Elasticity in Cloud Computing: State of the Art and Research Challenges

Yahya Al-Dhuraibi, Fawaz Paraiso, Nabil Djarallah, Philippe Merle

Elasticity in Cloud Computing: State of the Art and Research Challenges

Yahya Al-Dhuraibi, Fawaz Paraiso, Nabil Djarallah, Philippe Merle

Abstract—Elasticity is a fundamental property in cloud computing that has recently witnessed major developments. This article reviews both classical and recent elasticity solutions and provides an overview of containerization, a new technological trend in lightweight virtualization. It also discusses major issues and research challenges related to elasticity in cloud computing. We comprehensively review and analyze the proposals developed in this field. We provide a taxonomy of elasticity mechanisms according to the identified works and key properties. Compared to other works in literature, this article presents a broader and detailed analysis of elasticity approaches and is considered as the first survey addressing the elasticity of containers.

Index Terms—Elasticity; Cloud Computing; Auto-scaling; Resource provision; Scalability; Containers.

1 Introduction

Cloud computing has been gaining more popularity in the last decade and has received a great deal of attention from both industrial and academic worlds. The main factor motivating the use of cloud platforms is their ability to provide resources according to the customer's needs or what is referred to as elastic provisioning and de-provisioning. Therefore, elasticity is one of the key features in cloud computing that dynamically adjusts the amount of allocated resources to meet changes in workload demands [1].

Cloud providers generally use virtualization-based approaches to build their stack. Virtualization makes it possible to run multiple operating systems and multiple applications on the same server at the same time. It creates an abstract layer that hides the complexity of both hardware and software working environments. Cloud computing paradigm allows workloads to be deployed and scaled-out quickly through the rapid provisioning of the virtualized resources. This deployment is done through virtual machines (VMs). Virtualization is commonly implemented with hypervisors. A hypervisor is one of the virtualization techniques that allows multiple operating systems to share a single hardware host in a way that each operating system appears to have its own independent resources. VMware ESX, KVM, Xen, and Hyper-V are examples of the world-wide used hypervisors.

Container-based virtualization, called operating system virtualization, is another approach to virtualization in which the virtualization layer runs as an application within the operating system (OS). Containers are a lightweight solution that allows faster start-up time and less overhead [2]. Therefore, since virtualization is a central part of cloud computing that helps to improve elasticity, we discuss cloud elasticity in the context of both VMs and containers. In the literature, there exist various definitions, mechanisms, strategies, methods, and solutions for elasticity in both industrial and research works.

Elasticity has been explored by researchers from academia and industry fields. Many virtualization technologies, on which cloud relies, continue to evolve. Thus, tremendous efforts have been invested to enable cloud systems to behave in an elastic manner and many works continue to appear. Therefore, we are motivated to provide a comprehensive and extended classification for elasticity in cloud computing. This article focuses on most aspects of the elasticity and it particularly aims to shed light on the emerging container elasticity as well as the traditional VMs. Although many elasticity mechanisms have been proposed in the literature, our work addressing more broader classification of elasticity taxonomy. It is also the first survey that highlights elasticity of containers. The major contributions of this article are summarized as:

- First, we propose a precise definition of elasticity and we highlight related concepts to elasticity such as scalability and efficiency and approaches to measure elastic systems.
- Second, we provide an extended classification for the elasticity mechanisms according to the configuration, the scope of the solution, purpose, mode, method, etc. For example, when discussing the mode of elasticity that can be reactive or proactive to perform elasticity decisions, we discuss in depth each mode by classifying the mode into other subcategories and presenting works that follow the mode as shown in Table 1.
- Third, we discuss the existing container technologies and their relation to cloud elasticity. This article is the first work that discusses container elasticity in presenting many recent works from the literature.

The remainder of the article is organized as follows. Section 2 explains the elasticity concept, its related terms,

- *Y. Al-Dhuraibi is with Scalair company, Hem, France. E-mail: yaldhuraibi@scalair.fr*
- *F. Paraiso is with Inria Lille - Nord Europe, Villeneuve d'Ascq, France. E-mail: fawaz.paraiso@inria.fr*
- *N. Djarallah is with Scalair company, Hem, France. E-mail: ndjarallah@scalair.fr*
- *P. Merle is with Inria, Lille - Nord Europe, Villeneuve d'Ascq, France. E-mail: philippe.merle@inria.fr*

Manuscript received May 30, 2017.

its classical solution classifications and our new extended classification, the tools, and platforms that have been used in the experiments of the existing works in the literature. This section describes cloud elasticity solutions in the VMs. Next, Section 3 presents the concept of containerization, and how it could improve elasticity in cloud computing. It discusses the few existing papers on cloud elasticity when containers are used. Then, in Section 4, we present the main research challenges in elasticity and also the limits in the new trend of containerization. Section 5 discusses some related work. Finally, Section 6 concludes the paper.

2 ELASTICITY

In order to well understand the elasticity, we describe some related concepts, in addition to a new refined and comprehensive definition for elasticity. We propose a classification and taxonomy for elasticity solutions based on the characteristics: configuration, scope, purpose, mode, method, provider, and architecture. This classification is a result of thorough study and analysis of the different industrial and academic elasticity solutions. This classification provides a comprehensive and clear vision on elasticity in cloud computing. We then review the elasticity evaluation tools and platforms implemented in diverse works.

2.1 Elasticity definition and its related terms

There have been many definitions in the literature for elasticity [3], [4], [5], [1]. However, from our point of view, we define elasticity as the ability of a system to add and remove resources (such as CPU cores, memory, VM and container instances) "on the fly" to adapt to the load variation in real time. Elasticity is a dynamic property for cloud computing. There are two types of elasticity as shown in Fig. 1: horizontal and vertical. Horizontal elasticity consists in adding or removing instances of computing resources associated with an application. Vertical elasticity consists in increasing or decreasing characteristics of computing resources, such as CPU time, cores, memory, and network bandwidth.

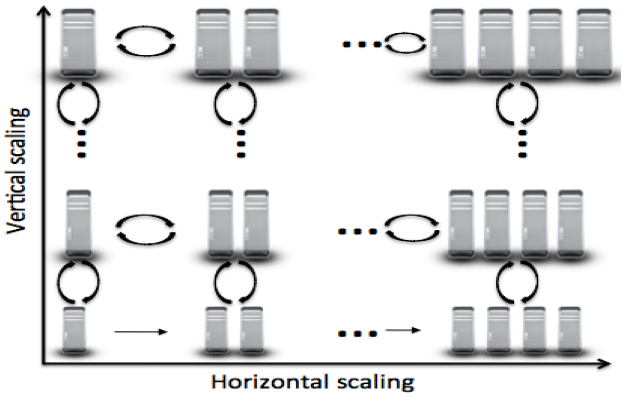

Fig. 1. Horizontal vs Vertical elasticity

There are other terms such as scalability and efficiency, which are associated with elasticity but their meaning is different from elasticity while they are used interchangeably in some cases. Scalability is the ability of the system to sustain increasing workloads by making use of additional resources [5], it is time independent and it is similar to the provisioning state in elasticity but the time has no effect on the system (static property). In order to have a complete understanding, we deduce the following equation that summarizes the elasticity concept in cloud computing.

$$Elasticity = \underbrace{scalability + automation}_{\text{auto-scaling}} + optimization$$

It means that the elasticity is built on top of scalability. It can be considered as an automation of the concept of scalability, however, it aims to optimize at best and as quickly as possible the resources at a given time. Another term associated with elasticity is the efficiency, which characterizes how cloud resource can be efficiently utilized as it scales up or down. It is the amount of resources consumed for processing a given amount of work, the lower this amount is, the higher the efficiency of a system. The amount of resources can relate to cost, power consumption, etc., depending on the targeted resource [6]. Generally, this is a measure of how well the application is using the provided resources. Higher cloud elastic system results in higher efficiency. The processes of resource provisioning and scheduling (i.e., jobs or customer' requests on instances) are both related to elasticity since they try to provision instances but in response to provider and customer tradeoffs. [7], [8] provision resources according to a utility model to satisfy customers' needs and a certain pricing model to increase service provider profit. The provisioning and scheduling processes may take a certain delay in order to meet SLAs and provider profit conditions.

It is worth noting that scaling up or down the resources can lead to a deviation of the current amount of allocated resources from the actual required resource demand. The accuracy of elasticity systems varies from one system to another. Over-provisioning and under-provisioning are two important factors that characterize an elastic system. The system enters in over-provisioning state once the resources provided (called supply S) are greater than the consumer required resources (called demand D), i.e., $S > D$. Though QoS can be achieved, over-provisioning state leads to extra and unnecessary cost to rent the cloud resources. Under-provisiong takes place once the provided resources are smaller than the required resources, i.e., $S < D$, and this causes performance degradation and violation of service level agreement (SLA). There is no common methodology to measure or determine temporal or quantitative metrics for elasticity. A consumer can measure the delay it takes to provision and deprovision resources, in addition to the sum of delays of over-provisioning and under-provisioning states to quantify different elastic systems [9].

[10] discusses methods to measure scalability and elasticity of a system. According to [10], effects of scalability are visible to the user via observable response times or throughput values at a certain system size. On the other hand, the elasticity, namely the resource resizing actions, may be invisible to the user due to their shortness or due to the transparency and dynamicity of resource provisioning. The effect of reconfiguration on performance metrics (e.g., response time) due to elastic adjustments of resources and the reaction time can quantify the elasticity. It is clear that

elasticity is controlled with time. Therefore, the speed is also very important in elasticity. Reaction time is the time interval between the instant when a reconfiguration has been triggered/requested and until the adaptation has been completed.

[11] proposes an approach for elasticity measurements. In addition to the over-provisioning and under-provisioning states, another state called just-in-need is introduced. Just-in-need denotes a balanced state, in which the workload can be properly handled and quality of service (QoS) can be satisfactorily guaranteed. The approach developed calculation formulas for measuring elasticity values based on the time intervals a system stays in one state. There are three states: over-provisioning, under-provisioning, and just-in-need. A set of rules is used to determine the state of a system based on the workload and computing resources. The equations can be obtained and calculated by directly monitoring the system or by using continuous-time Markov chain (CTMC) model. The drawback of the proposed system is that it assumes the system is in a certain state based on rules. For example, the system is in just-in-need state if the number of requests (j) is greater than the number of VMs (i) and less than 3 multiplied by the number of VMs (i), i.e., $(i < j \leqslant 3i)$. We cannot guarantee the certainty for these rules on all elastic systems.

2.2 Elasticity taxonomy

Elasticity solutions build their mechanisms on different strategies, methods, and techniques. Therefore, different classifications [3], [4], [12], [13], [14] have been proposed according to the characteristics implemented in the solutions. We have investigated many industrial and academic solutions, in addition to papers in the elasticity literature, and then we propose the classification shown in Fig. 2. It is an extended and complementary elasticity classification as compared to classification in [3], [4], [12], [13], and [14].

Next subsections explain in details each characteristic and mechanism used. The solutions are classified according to the chosen *configuration*, *scope*, *purpose*, *mode or policy*, *method or action*, *architecture*, and *provider*.

2.2.1 *Configuration*

Generally, configuration represents a specific allocation of CPU, memory, network bandwidth and storage [15]. In the context of our classification (see Fig. 2), configuration represents the method of the first or initial reservation of resources with a cloud provider. During the first acquisition of resources, the consumer either chooses from a list of offer packs or specifies its needs, i.e., combining different resources. Therefore, the configuration can be either rigid (fixed) or configurable. The rigid mode means that the resources are offered or provisioned in a constant capacity. The virtual machine instances (VMIs) are found with a predefined resource limit (CPU, Mem, etc.) called instances such as Amazon EC2 (offering 38 instances), Microsoft Azure (offering many series A, D, DS, G, and GS and each series has different VM sizes). In the cloud market, the VMIs are offered in various configurations.

The problem with rigid configuration is that the resource rarely meets the demand, therefore, there is always under-provisioning or over-provisioning. The configurable mode allows the client to choose the resource such as number of CPU cores in the VMs. ProfitBricks [16] and CloudSigma [17] are examples of this type.

The customers can reserve the resources according to the following reservation methods [15]:

- **On-demand reservation**: The resources are reserved immediately or the requests will be rejected if there are no enough available resources.
- **In advance reservation**: The clients send initial requests to reserve resources and a fixed price charge is required to initiate the reservation, the resources must be available at a specific time.
- **Best effort reservation**: Reservation requests are queued and served accordingly such as Haizea, an open-source VM-based lease management architecture used in OpenNebula [18].
- **Auction-based reservation**: Specific resource configurations are reserved dynamically as soon as their prices are less than bid amount offered by the customer [19].
- There are other types of reservation such as Amazon's scheduled reserved instances, Amazon's dedicated instances, Google's preemptible instances, etc.

2.2.2 *Scope*

The elasticity actions can be applied either at the infrastructure or application/platform level. The elasticity actions perform the decisions made by the elasticity strategy or management system to scale the resources.

When the elasticity action control is in the application or platform level, it is named embedded elasticity and this will be described below. Google App Engine [20], Azure elastic pool [21] are examples of elastic Platform as a Service (PaaS). The applications can be either one tier or multi-tiers, most of the existing elasticity solutions are dedicated to one-tier applications where elasticity management is performed for one tier only, mostly the business tier. However, there are some recent works that perform elasticity actions on multi-tier applications such as [22], [23], [24], [25], [26], [27], [28], [29].

Beside this, the elasticity actions can be performed at the infrastructure level where the elasticity controller monitors the system and takes decisions. The cloud infrastructures are based on the virtualization technology, which can be VMs or containers. Most of the elasticity solutions [29], [30], [31], [32], [33], [34], [35], [36], [37], [38] are dedicated to the infrastructure level, and these solutions are suitable for client-server applications. However, other elastic solutions exist for the other types of applications. For example, [39] and Amazon EMR are elastic solutions for MapReduce applications, [40] describes an elasticity solution for streaming applications, while [12] discusses approaches for elasticizing scientific applications. Due to the nature of a scientific application such as parallelism, models (e.g., serial, multi-thread, single program multiple data, master-worker, etc.), an elasticity solution can not be generalized for scientific applications. The elasticity solution must consider the internal structure and behavior of a scientific application, therefore, to have a reliable elastic solution, it should be embedded in the application source code. It is worth mentioning that

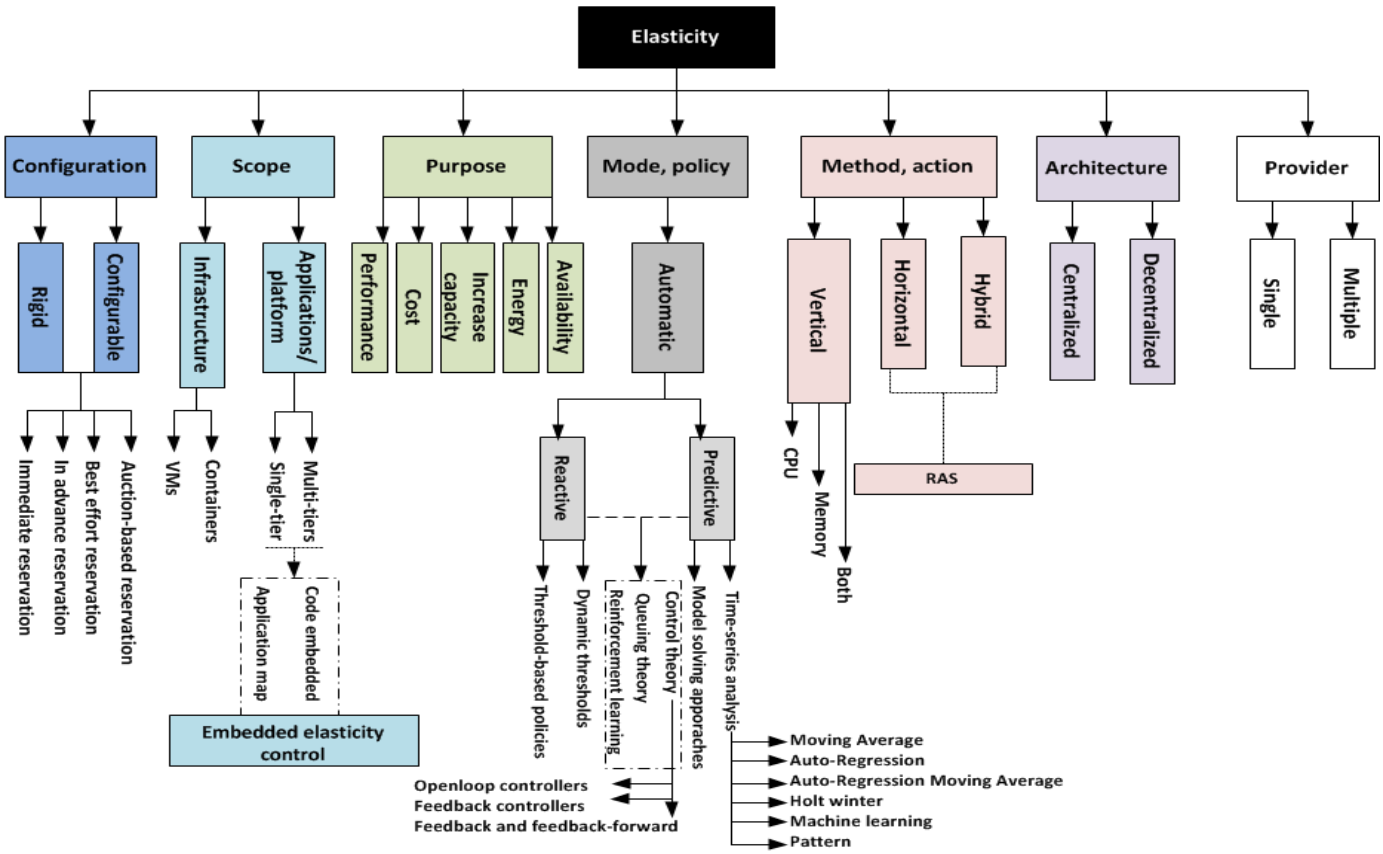

Fig. 2. Classification of the elasticity mechanisms

some elasticity controllers support sticky sessions. The session is the concept of a series of interactions between a client and the application. The stateful nature of some sessions forces the user to be connected to the same server each time he submits a request within the session, if the session data is stored in the server, such sessions are called sticky sessions. Sticky sessions cause issues on efficiently utilizing elastic resources because they limit the ability of the elastic controller to terminate under-utilized instances when there are still unfinished sessions handled by them. Most solutions support stateless applications, while few solutions [41], [42] handle stateful instances or sticky sessions.

Embedded Elasticity

Most of the existing solutions are dedicated to server-based applications. However, there are many different application modules that have different execution behavior particularities such as scientific applications. Therefore, we named these types of solutions as embedded elasticity controller. In the embedded elasticity, elastic applications are able to adjust their own resources according to runtime requirements or due to changes in the execution flow. There must be a knowledge of the source code of the applications. As seen in Fig. 2, we classify these solutions into two subcategories.

- **Application Map**: The elasticity controller must have a complete map of the application components and instances. As it is well known that some applications comprise of many components and each component may have many instances. These components are either static or dynamic. Static components must be launched once the application starts, while dynamic components can be started or stopped during the application runtime. In addition, there are interconnections between these instances. Therefore, the elasticity controller must have all the information about the application instances, components, and interconnections that allow it to perform elasticity actions for applications. [43], [44], [45], [46] are examples of such works.
- **Code embedded**: The idea here is that the elasticity controller is embedded in the application source code. The elasticity actions are performed by the application itself. While moving the elasticity controller to the application source code eliminates the use of monitoring systems, there must be a specialized controller for each application. Examples of these solutions are [12] and [39].

2.2.3 *Purpose*

Elasticity has different purposes such as improving performance, increasing resource capacity, saving energy, reducing cost and ensuring availability. Once we look to the elasticity objectives, there are different perspectives. Cloud IaaS providers try to maximize the profit by minimizing the resources while offering a good Quality of Service (QoS), PaaS providers seek to minimize the cost they pay to the cloud and the customers (end-users) search to increase their Quality of Experience (QoE) and to minimize their payments. QoE is the degree of delight or annoyance of the user of an application or service [47]. The goal of QoE management is then to deliver the cloud application

to the end user at high quality, at best while minimizing the costs of the different players of the cloud computing stack (IaaS, PaaS, SaaS) [48]. As consequences, there have been many trade-offs. Elasticity solutions cannot fulfill the elasticity purposes from different perspectives at the same time, each solution normally handles one perspective. However, some solutions try to find an optimal way to balance some of the contradicted objectives. [8] scales resources according to a utility model to reply to customers QoE and a dedicated pricing model to increase service provider gains. [49] presents a survey of how to look for balancing two opposed goals, i.e., maximizing QoS and minimizing costs. As shown in Table 1, most proposals improve the performance. However, there are other works that have described the use of elasticity for purposes, such as, increasing the local resources capacity [32], [50], cost reduction [22], [23], [37], [51], [52], [53], [54], [55], [56], [57], [58], [59] and energy savings [38], [44], [53], [60], [61]. Many of the elasticity management solutions as indicated in [13] takes into consideration Quality of Business metrics that are often expressed in monetary units and include service price, revenue per user, revenue per transaction, provisioning cost, and budget. Examples of the solutions that ensure the availability include [24], [28], [53], [62], [63]. [61] takes into consideration both the provider profit and user QoE. In this work, various algorithms have been studied in order to obtain the best trade-off between the user or SLA requirements and provider profit. [64], [65] also proposes QoE-aware management elastic approaches that try to maximize users' satisfaction without extra costs. Other examples for improving the performance are found in the research community and commercial clouds such as Rackspace [33], Scalr [36], RightScale [34].

2.2.4 *Mode or policy*

Mode (policy) refers to the needed interactions (or manner) in order to perform elasticity actions. Elasticity actions are performed by an automatic mode. Scaling actions can be achieved by manual intervention from the user. As indicated in [12], there is also another mode, which is called programmable mode. In fact, it is just the same as manual mode because the elasticity actions are performed using API calls. Though a cloud provider offers an interface which enables the user to interact with the cloud system. The manual policy is used in some cloud systems such as Datapipe [148], Rackspace [33], Microsoft Azure [21], and the Elastin framework [45] where the user is responsible for monitoring the virtual environment and applications, and for performing all scaling actions. This mode can not be considered as an elasticity mode since it violates the concept of automation.

Automatic mode: All the actions are done automatically, and this could be classified into reactive and proactive modes.

1) **Reactive mode** means the elasticity actions are triggered based on certain thresholds or rules, the system reacts to the load (workload or resource utilization) and triggers actions to adapt changes accordingly. Most cloud platforms such as Amazon EC2 [30], Scalr [36], Rightscale [34] and other research works such as [55], [122], [149], [150] use this technique.
   - **Static thresholds** or role-condition-actions: The elasticity actions are fired to scale up or down the resources when the role-condition is met. This policy depends on thresholds or SLA requirements, the conditions are based on the measurements of one or a set of metrics such as CPU utilization, memory utilization, response time, etc. Two or more thresholds are used for each performance metric. The measured metrics are compared against fixed thresholds. For example, if CPU utilization is greater than 80%, and this situation lasts 5 minutes, then the resource is scaled up. Amazon EC2, Rightscale and other research works such as [24], [27], [28], [32], [54], [55], [57], [63], [67], [78], [80], [83], [84], [95], [96], [97], [98] use such mechanism.
   - **Dynamic thresholds**: Previous thresholds are static and are fixed user-defined values. On the contrary, dynamic thresholds, called adaptive thresholds, changed dynamically according to the state of the hosted applications. The works in [58], [60], [100], [101] use the adaptive utilization thresholds technique. The thresholds such as CPU utilization are changed dynamically.
2) **Proactive mode**: This approach implements forecasting techniques, anticipates the future needs and triggers actions based on this anticipation. Many academic works such as [31], [35], [37] use this mode as we will see in the following proactive techniques.
   - **Time series analysis**: Time series is a sequence of measurements taken at fixed or uniform intervals [151]. Time series analysis is used to identify repeating patterns in the input workload and to attempt to forecast the future values. In other terms, time series analysis is responsible for making an estimation of the future resource and workload utilization, after this anticipation, the elasticity controller will perform actions based on its policy (e.g., a set of predefined rules). Generally, the time series analysis has two main objectives. Firstly, predicting future values (points) of the time series based on the last observations (recent usage). Secondly, identifying the repeated patterns, if found, then use them to predict future values. The recent history window (resource usage) is used as input to the anticipation technique which in turn generates future values. For achieving the first objective, there are several techniques such as Moving-Average, Auto-Regression, ARMA, Holt winter and machine learning. For example, [29], [51], [79], [82], [83], [84], [87], [90], [110], [111] use machine learning. [27], [106], [107] use Moving-Average. [23], [24], [25], [108], [109] follow Auto-Regression technique while [23], [35], [67] follow ARMA approach. Holt winter is used by [54], [107]. In order to achieve the second purpose, various techniques are used to inspect the repetitive patterns in time series: pattern matching [108], [112], Fast Fourier Transform (FFT) [108], auto correlation [152], histogram [108].
   - **Model solving mechanisms** are approaches based on probabilistic model checking or mathematical modeling frameworks to study the diverse behaviours of the system and anticipate its future states such as Markov Decision Processes (MDPs), probabilistic

TABLE 1
Examples on elasticity solutions

Elasticity	**Configuration**	**Rigid**	[21], [30]			
		Configurable	[16], [17]			
	Scope	**Infrastructure**	**VMs**	[29], [30], [31], [32], [33], [34], [35], [36], [37], [38]		
			Containers	[56], [66], [67], [68], [69], [70]		
		Application/ Platform	**Single-tier**	[40], [44], [45], [46], [50], [71], [72], [73], [74]		
			Multi-tier	[22], [23], [24], [25], [26], [27], [28], [29], [75]		
			Application map	[43], [44], [45], [46]		
			Code embedded	[12], [39]		
	Purpose	**Performance**	[22], [23], [26], [27], [29], [30], [31], [32], [33], [34], [35], [36], [38], [40], [45], [46], [55], [58], [67], [71], [73], [75], [76], [77], [78], [79], [80], [81], [82], [83], [84], [85], [86], [87], [88], [89], [90], [91], [92]			
		Cost	[22], [23], [37], [51], [52], [53], [54], [55], [56], [57], [58], [59], [93], [94]			
		Capacity	[32], [50]			
		Energy	[38], [44], [53], [60], [61]			
		Availability	[24], [28], [53], [62], [63]			
	Mode	**Automatic**	**Reactive**	**Threshold-based policies**	[24], [27], [28], [32], [54], [55], [57], [63], [67], [78], [80], [83], [84], [91], [95], [96], [97], [98], [99]	
				Dynamic thresholds	[58], [60], [100], [101]	
				Reinforcement learning	[72], [96], [102]	
				Queuing theory	[55], [57], [59], [103]	
				Control theory	[89], [100], [101], [104], [105]	
			Proactive	**Time series analysis**	**Moving average**	[27], [106], [107]
					Auto regression	[23], [24], [25], [108], [109]
					ARMA	[23], [35], [67]
					Holt winter	[54], [107]
					Machine learning	[29], [51], [79], [82], [83], [84], [87], [90], [110], [111]
					Pattern	[108], [112]
				Model solving approaches	[75], [92], [99]	
				Reinforcement learning	[113], [114]	
				Queuing theory	[103], [114], [115], [116], [117]	
				Control theory	[22], [26], [91], [105], [118]	
	Method	**Horizontal scaling**	[23], [24], [28], [29], [30], [32], [33], [34], [35], [36], [50], [55], [63], [71], [79], [81], [99], [119], [120], [121], [122], [123], [124]			
		Vertical scaling	**CPU**	[26], [38], [116], [125], [126]		
			Memory	[74], [127], [128]		
			CPU & Mem.	[31], [54], [129], [130], [131], [132], [133]		
		Migration	[38], [45], [61], [73], [86], [90], [134], [135], [136], [137], [138], [139], [140], [141]			
		Hybrid	[37], [38], [44], [51], [58], [90], [91], [142]			
	Architecture	**Centralized**	Most approaches presented in this table, except the decentralized ones.			
		Decentralized	[93], [94], [143], [144], [145], [146]			
	Provider	**Single**	[22], [23], [24], [25], [27], [54], [55], [57], [78], [79], [82], [87], [100], [147]			
		Multiple	[28], [53], [62], [63], [66], [93], [142], [145]			

timed automata (PTAs). [92] and [99] are examples of works that adopt model solving approaches. [75] is a more recent work that uses Alloy models to increase the performance of the model solving (i.e., most of the MDP models and combinations are built offline using a formal specification in Alloy which eliminates the runtime overhead of MDP construction for each adaptation decision).

There are other mechanisms that can be used with both reactive and proactive approaches (when accompanied with other mathematical models such as Markov Decision Process, Q-learning algorithm, Model predictive control (MPC)):

- **Reinforcement Learning (RL)** is a computational approach that depends on learning through interactions between an agent and the system or environment. The agent (decision-maker) is responsible for taking decisions for each state of the environment, trying to maximize the return reward. The agent learns from the feedback of the previous states and rewards of the system, and then it tries to scale up or down the system by choosing the right action. For example, [72], [96], [102] use RL in reactive mode while [113], [114] use RL in proactive mode.
- **Control theory** controls the system functions in reactive mode [89], [100], [101], [104], [105], but there are some cases in which they can work in proactive mode [22], [26], [105], [118]. There exist three types of these controllers: Openloop controllers, Feedback controllers, and Feedback and Feedback-forward controllers. Openloop (non feedback) controllers compute the input to the system, these controllers do not have feedback to decide whether the system is working well or not. Feedback controllers monitor the output of the system and correct the deviation against the desired goal. Feedback-forward controllers predict errors in the output, anticipate the behavior of the system and react before errors occur. Feedback and feedback-forward controllers are usually combined.
- **Queuing theory** is a mathematical study for queues

in the system taking in consideration the waiting time, arrival rate, service time, etc. Queuing theory is intended for systems with a stationary nature. It can be used to model applications (single or multi-tiers). [55], [57], [103], [59] use queuing theory in reactive mode while [103], [114], [115], [116], [117] adhere to the queuing theory principles in predictive mode. For example, [115] proposes a model that estimates the resources required for a given workload λ, the mean response time, and other parameters. [116] uses queue length and inverse model to anticipate capacity requirement taking into consideration also the target response time.

Before finishing this section, it is worth mentioning that many works generally span across different subcategories, use more than one technique and that is why they appear more than once in Table 1. Many systems and proposals adhere to use a combination of reactive and proactive policies, e.g., [83], [84] use threshold and machine learning policies. [27] implements threshold-based rules and moving average while [24] uses thresholds and auto-regression. [54] uses thresholds and holt-winter. [55], [57] combine thresholds based rules and queuing theory in reactive mode only. Similarly, [100], [101] use dynamic thresholds and queuing theory while [96] combines thresholds and enforcement learning. [67] uses static thresholds for CPU and memory usage, ARMA to predict the number of requests for Web applications. Other works used more than one technique in proactive mode. For example, [23] implements auto-regression and ARMA. [114] uses reinforcement learning and queuing theory. [108] combines auto-regression and pattern matching.

2.2.5 *Method*

To deploy the elasticity solutions, one or hybrid of the following methods is implemented: horizontal scaling, vertical scaling. Horizontal elasticity allows adding new instances while vertical elasticity, referred to as fine-grained resource provisioning, allows resizing the resources of the instance itself to cope with the runtime demand. The instances can be VMs, containers, or application modules. Horizontal and vertical techniques have their advantages and shortcomings. Horizontal elasticity is simple to implement and it is supported by hypervisors. It has been widely adopted by many commercial providers. However, horizontal elasticity can lead to inefficient utilization of the resources due to the fact that it provides fixed or static instances, which sometimes cannot fit exactly with the required demand. On the contrary, vertical elasticity allows resizing the instances but it is not fully supported by all hypervisors, although new hypervisors such as Xen, VMware support it.

- **Horizontal scaling** is the process of adding/removing instances, which may be located at different locations. Load balancers are used to distribute the load among the different instances. It is the most widely implemented method, most cloud providers such as Amazon [30], AzureWatch [71], and many other academic works as shown in Table 1 use this method.
- **Vertical scaling** is the process of modifying resources (CPU, memory, storage or both) size for an instance at runtime. It gives more flexibility for the cloud systems to cope with the varying workloads. There are many works [26], [116], [38], [125], [126] that only focus on CPU vertical resizing, other works [74], [127], [128] focus on memory resizing. It is worth noting that, there have been many techniques used in literature for memory resizing such as EMA, page faults, ballooning [132]. While there exist some proposals [31], [129], [130], [131], [133] that control both resources (CPU, memory). [132] is a particular work that not only controls both resources (CPU, memory) but also coordinates the degree of vertical resizing of the CPU in relation to the memory. [54] proposes a mechanism to resize CPU, Disk, and memory. ProfitBricks and RightScale cloud providers offer this feature to their customers.

  Migration can be also considered as a needed action to further allow the vertical scaling when there is no enough resources on the host machine. However, it is also used for other purposes such as migrating a VM to a less loaded physical machine just to guarantee its performance, etc. Several types of migration are deployed such as live migration [38], [45], [86], [90], [139] and no-live migration [153]. Live migration has two main approaches post-copy [141] and pre-copy [134]. Post-copy migration suspends the migrating VM, copies minimal processor state to the target host, resumes the VM and then begins fetching memory pages from the source [154]. In pre-copy approach, the memory pages are copied while the VM is running on the source. If some pages changed (called dirty pages) during the memory copy process, they will be recopied until the number of recopied pages is greater than dirty pages, or the source VM will be stopped, and the remaining dirty pages will be copied to the destination VM.

Before performing migration or replication, a Resource Allocation Strategy (RAS) [155] is used. RAS decides where the destination or new instance will be allocated or created, on which server, on which cloud data center. RAS is based on cost and speed of VM, the CPU usage of the physical machine, the load conditions specified by the user, the maximum profit [155], etc.

Many works have used a combination of the previously described methods. [37], [44] proposals implement replication and migration methods. [58] proposes an approach that creates new small replicas and then attaches them to load balancer or deploys a new big server and removes the previous server. The application is then reconfigured to use the provided new resources. [51] proposes a framework that uses a combination of vertical resizing (adding resources to existing VM instances) or horizontal scaling (adding new VM instances). [90] reconfigures CPU and memory, live migration is triggered when there is no sufficient resources. [38] configures CPU voltage and frequency and it also uses live migration.

2.2.6 *Architecture*

The architecture of the elasticity management solutions can be either centralized or decentralized. Centralized architecture has only one elasticity controller, i.e., the auto-scaling system that provisions and deprovisions resources. Most solutions presented in the academic literature and

business world have a centralized architecture while there are some solutions that are decentralized such as [143] and [144]. In decentralized solutions, the architecture is composed of many elasticity controllers or application managers, which are responsible for provisioning resources for different cloud-hosted platforms. In addition to an arbiter which is the key master component in a decentralized approach because it is charged to allocate resources to the other controllers at the different system components. Multi-Agent Systems (MAS) also represent a distributed computing paradigm based on multiple interacting agents. The interacting agents with cloud shape a new discipline called agent-based cloud computing. Multiple agents allow cloud computing to be more flexible and more autonomous [156]. MAS technologies have been used to decentralize the elasticity management decision [157]. Some examples of existing works using MAS for cloud elasticity, cloud service reservation, and SLA negotiation include [94], [93], [145], [146].

2.2.7 *Provider*

Elastic solutions can be applied to a single or multiple cloud providers. A single cloud provider can be either public or private with one or multiple regions or datacenters. Multiple clouds in this context means more than one cloud provider. It includes hybrid clouds that can be private or public, in addition to the federated clouds and cloud bursting. Most of the elasticity solutions and proposals support only a single cloud provider. However, there are other works [28], [53], [62], [63], [142] that handle elasticity between multiple cloud providers simultaneously.

2.3 Elasticity performance evaluation

Experiments are very important for the performance evaluation of elastic cloud systems. However, there is no standard method for evaluating auto-scaling and elasticity techniques due to the uncertainties in the workloads and unexpected behaviors of the system. Therefore, researchers use different testing environments according to their own needs. We introduce the common experimental platforms, workloads, and application benchmarks, as shown in Fig. 3, that have been used in the literature.

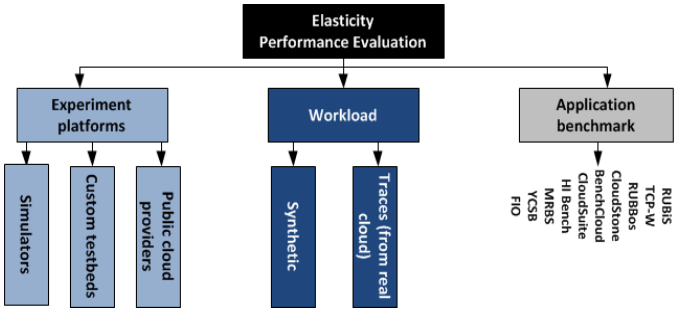

Fig. 3. Performance evaluation tools

2.3.1 *Experimental platforms*

Experiments can be achieved using simulators, custom testbeds or real cloud providers.

Simulators are widely used to simulate cloud platforms [152]. Using simulators in evaluating elasticity systems and application behaviors offer significant benefits, as they allow developers to test the performance of their systems in a repeatable and controllable free of cost environment and they also allow to tune the performance bottlenecks before real-world deployments on commercial clouds. Some cloud simulators are:

- CloudSim [158]: a powerful framework for modeling and simulation of cloud computing infrastructures and services. It is widely used in research works.
- ContainerCloudSim [159] is another simulation tool that integrates most functionalities of CloudSim. It aims to provide support for modeling and simulation of containerized cloud computing environments. It supports modeling and simulation for container resource management, placement, migration on the simulated cloud environment.
- GreenCloud [160]: a framework used to develop novel solutions in monitoring, resource allocation, workload scheduling, as well packet-level simulator for energy-aware cloud computing data centers.
- OMNeT++ [161]: a framework used primarily for building network simulators but it is also used for cloud platforms.
- iCanCloud [162]: targeted to conduct large experiments, provides a flexible and customizable global hypervisor for integrating any cloud brokering policy.
- SimGrid [163]: a simulator for large-scale distributed systems such as clouds.
- EMUSIM [164]: an integrated emulation and simulation environment for modeling, evaluation, and validation of the performance of cloud computing applications.

Custom testbeds offer more control on the platform, but they require extensive efforts for system configuration. For deploying custom testbeds or clouds, many technologies are used such as hypervisors (Xen, VMWare ESXi, KVM, etc.), cloud orchestrators such as OpenStack, CloudStack, OpenNebula, Eucalyptus, and the commercial VCloud. Academic cloud testbeds such as Grid5000, FutureGrid, open research clouds are also widely used.

Public clouds. While achieving experiments on a real cloud reflects the reality, it has a big drawback: there are external factors that cannot be controlled, which could impact negatively the tested system. In addition, a cloud provider offers the infrastructure (on which the experiment will be launched), but monitoring and auto-scaling system, application benchmark, workload generators are still needed.

2.3.2 *Workloads*

User requests or demand together with timestamps are required for the tested platforms (to derive the experiments). Workloads can be synthetic or real.

- **Synthetic workloads** are generated with special programs in a form of different patterns. Faban, JMeter, httperf, Rain are examples of workload generators.
- **Real workloads** are obtained from real cloud platforms and stored into trace files. World cup [165], Clark

net [166], and Google Cluster trace [167] are examples of real workloads. Different application workloads have different characteristics. Therefore, there exists no single elasticity algorithm which is perfect for the diverse types of workloads. Workload analysis and classification tools [27], [168] are used to analyze workloads and assign them to the most suitable elasticity controller based on the workload characteristics and business objectives.

2.3.3 *Application benchmark*

To test the scale up/down and scale out/in capabilities of a cloud platform, a set of cloud benchmarks are widely used. Benchmarks are commonly used to evaluate the performance and scalability of the servers [152]. Experiments are conducted mainly on all cloud platforms and models including IaaS, PaaS, SaaS, etc. Benchmarks have both applications and generators. RUBBos [169], RUBiS [170], TCP-W [171], CloudStone [172], YCSB [173], MRBS [174] and FIO [175], BenchCloud at USC, CloudSuite [176], and HI Bench [177], are well-known benchmarking platforms.

3 Containerization

This section discusses container technologies, their pros and cons. We then present the concepts and surrounding technologies behind containers. Finally, we discuss works from literature related to elasticity of containers.

3.1 Pros and Cons

Hypervisors are the most widely used virtualization techniques in cloud computing. However, with the need of more flexibility, scalability, and resource efficiency, cloud providers are tapping hands-on into containers [178]. Containers or what is referred to as operating system-level virtualization have evolved dramatically. Container-based virtualization is much more lightweight and resource efficient than VM-based virtualization. Containers isolate processes on the core-level of the OS. In other words, they share the same OS and they do not need guest OS, which allows to manage resources efficiently and have more instances on the same server. The use of containers eliminates the hypervisor layer, redundant OS kernels, libraries, and binaries, which are needed to run workloads or applications in a virtual machine with the classical hypervisor virtualization. On the contrary, the traditional hypervisor virtualization requires a full OS on the VM, which consumes resources and causes an extra overhead. Fig. 5 compares application deployment using a hypervisor and a container manager. As shown in Fig. 5, the hypervisor-based deployment requires different operating systems and adds an extra layer of virtualization compared to containerization.

Container technologies provide some advantages such as:

- Containers decrease the start up time, processing and storage overhead when compared to the traditional VMs [179].
- Containers isolate and control processes and resources. Namespaces provide an isolation per process. In Linux OS, cgroups isolate resource usage such as memory, CPU, block I/O and provide resource management. Namespaces and cgroups do not incur overhead or performance penalty.
- Containers solve the issues of portability and consistency between environments [180].

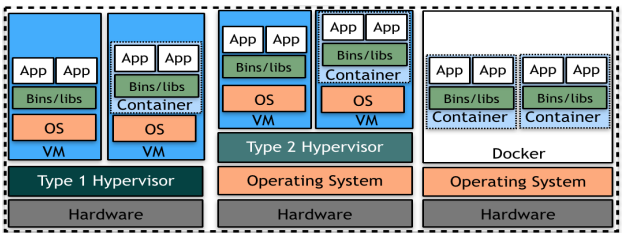

Fig. 4. Container-based Virtualization vs. Traditional Virtualization

While containerization technology offers many advantages, it has the following shortcomings:

- The use of containers poses security implications. The user processes are isolated on the shared OS but it is hard, at least until now, to provide the same level of isolation between containers as VMs do.
- Since development of new container managers such as Docker is recent, it lacks many functionalities. The development is in progress in this attractive domain.
- New container standards support only 64 bit systems.

3.2 Container technologies

The concept of containers has existed for over a decade. Mainstream Unix-based operating systems, such as Solaris, FreeBSD, Linux, had built-in support for containers. The interest in containers led to many actors to develop solutions. There are various implementations of containers such as:

Docker [181] is an open source management tool for containers that automates the deployment of applications. Docker uses a client-server architecture and it consists of three main components: *Docker client*, *Docker host* and *Docker registry*. Docker host represents the hosting machine on which Docker daemon and containers run. Docker daemon is responsible for building, running, and distributing the containers. Docker client is the user interface to Docker.

Rocket (rkt) is an emerging new container technology. With the advent of CoreOS [182], a new container called Rocket is introduced. Besides rkt containers, CoreOS supports Docker. Rocket was designed to be a more secure, interoperable, and open container solution. Rocket is a new competitor for Docker.

LinuX Containers (LXC) [183] is an operating system-level virtualization method for running multiple isolated Linux systems. It uses kernel-level namespaces to isolate the container from the host.

LXD [184] is a lightweight hypervisor, designed by Canonical, for Linux containers built on top of LXC to provide a new and better user experience. LXD and Docker make use of LXC containers.

Others: there are other open source light virtualization technologies such as BSDJail [185] and OpenVZ [186].

Docker and Rocket are the most recent used container technologies due to their enhanced features. We present

some of their surrounding technologies [187] in Fig. 5. Docker uses `runc` and `libcontainer` runtimes that enable interactions with Linux kernel components (cgroups, namespaces) to create and control containers. Rocket uses rkt and CoreOS runtimes. For the management, Docker uses Docker Engine that includes both Docker daemon and Docker client for interacting with Docker daemon. Docker daemon provides an API that abstracts container control functions. Rkt CLI is the container management functionality in Rocket. Docker containers can be defined using Docker images where container instances are created from these images. The images are created with Dockerfiles, text files containing all the commands needed to build Docker images. Rkt supports Docker images, as well as Application Container Images (ACI). Docker registry is the service responsible for storing and distributing images.

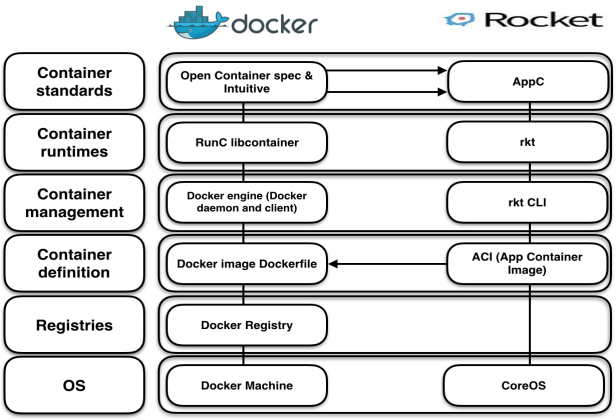

Fig. 5. Docker and Rocket Technologies

3.3 Container orchestration and management tools

Container adoption is expected to grow across all application life cycle steps, especially the production phase. However, some applications typically deal with workloads that have dozens of containers running across multiple hosts. This complex architecture dealing with multiple hosts and containers demands a new set of management tools.

Docker Swarm is a well-known clustering management tool for Docker containers. Swarm makes use of the Docker standard interface (API) in order to achieve its tasks such as starting Docker, choosing host to run containers on. Swarm consists of Swarm agents and a Swarm manager. Swarm agents are run on each host, the manager orchestrates, schedules containers on the hosts. Swarm uses discovery process to add hosts to the cluster, and it supports both Rocket and Docker containers. Swarm uses Docker-compose to support horizontal elasticity.

Kubernetes is another powerful container orchestration tool built by Google [188]. Kubernetes has brought new concepts about how containers are organized and networked. Along with managing single containers, it manages pods. Pod is a group of containers that can be created, deployed, scheduled and destroyed together. Kubernetes supports flat networking space, containers in a pod share the same IP, where pods can talk to each other without the need for NAT. In Kubernetes, replication controllers are responsible for controlling, and monitoring the number of running pods (called replicas) for a service [189], when a replica fails, a new one will be launched, and this improves reliability and fault tolerance. Kubernetes supports horizontal elasticity via its internal Horizontal Pod Autoscaling (HPA) system. HPA allows to automatically scale the number of pods based on observed CPU utilization. It uses reactive threshold-based rules for CPU utilization metric [190].

CoreOS Fleet is a cluster management tool that represents the entire cluster as a single init system [191]. Fleet is a low-level cluster management tool that allows a higher-level solution such as Kubernetes to be settled on the top. It provides a flexible management for the containers: fleet can start, stop containers, get information about the running services or containers in the different machines of the cluster, migrate containers from one host to another. It is designed to be fault-tolerant, and it supports both Rocket and Docker containers.

Apache Mesos [192] is an open-source cluster manager designed to manage and deploy application containers in large-scale clustered environments. Mesos, alongside with a job system like Marathon, takes care of scheduling and running jobs and tasks. It also supports horizontal elasticity.

OpenStack Magnum is a project that facilitates the utilization of container technology in OpenStack. It adds multi-tenant integration of prevailing container orchestration software for use in OpenStack clouds.

Fig. 6 shows some of the most used orchestration tools that are used to run applications on a distributed cluster of machines. These tools use service discovery such as etcd, Zookeeper, or Consul to distribute information between services or cluster hosts.

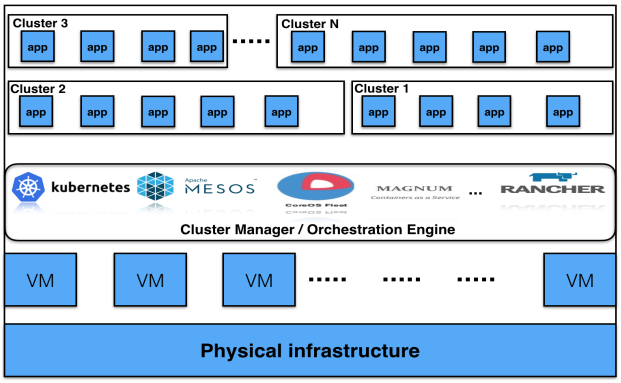

Fig. 6. Container orchestration engines

3.4 Elasticity of containers

Although containers are gaining wide-spread popularity among cloud providers, there are few works addressing elasticity of containers. As shown in Fig. 2, the elasticity solutions utilize various policies and methods. They have different purposes, configurations, and architectures. These mechanisms applied to the VMs can also be applied to containers as described below.

[68] proposes a design of a system used for developing and automatically deploying micro services. The proposed approach manages more instances of the application when

load increases and scales down/in for fewer demands to conserve energy. The requests count and memory load are monitored, when they arrive a certain threshold, containers are scaled out or in. A replication method is used to achieve horizontal elasticity of container instances.

[56] proposes a control architecture that dynamically and elastically adjusts VMs and containers provisioning. In this work, containers and VMs can be adjusted vertically (by varying the computational resources available for instances) or horizontally (change the number of instances) according to an objective function that searches to minimize costs.

[66] proposes a framework called MultiBox. MultiBox is a means for creating and migrating containers among cloud service providers. MultiBox makes use of the Linux cgroups to create and migrate containers that are isolated from the rest of the host OS. MultiBox containers support both stateful and stateless applications.

[193] proposes an approach for the application live migration in Linux container for better resource provisioning and interoperability. This approach uses Checkpoint/Restore In Userspace (CRIU) [194], a Linux functionality that allows container live migration.

Promox VE [70] also permits manual vertical resizing and migration for the LXC [183] and OpenVZ [186] containers. Promox VE is an open source server virtualization management software.

DoCloud [67] is an elastic cloud platform based on Docker. It permits to add or remove Docker containers to adapt Web application resource requirements. In DoCloud, a hybrid elasticity controller is proposed that uses proactive and reactive models to scale out and proactive model to scale in. Since cloud elasticity with containers is in its infancy, almost all the elasticity actions in containers elasticity solutions are performed using reactive approach that is based on pre-defined thresholds. However, DoCloud uses dynamic re-dimension method or predictive approaches to trigger elasticity actions. It uses a hybrid reactive, proactive controller that adopts threshold and ARMA approaches.

[69] proposes a model-driven tool to ensure the deployability and the management of Docker containers. It allows synchronization between the designed containers and those deployed. In addition, it allows to manually adjust container's vertical elasticity.

[91] proposes a horizontal and vertical auto-scaling technique based on a discrete-time feedback controller for VMs and containers. This novel framework allows coordinating infrastructure and platform adaptation for web applications. The application requirements and metadata must be precisely defined to enable the system to work. It inserts agents for each container and VM for monitoring and self-adaptation.

As described in the works related to container elasticity, containers can be scaled horizontally and vertically. However, in order to implement the mechanisms used in VMs, some modifications are needed. For example, in reactive approaches, breath duration is a period of time left to give the system a chance to reach a stable state after each scaling decision, since containers adapt very quickly to workload demand, breath duration must be small when compared to VM. To our knowledge, there is no work that adopts proactive approaches to scale containers except [67] which uses ARMA prediction. In addition, container adaptations, its hosted application adaptations and the monitoring system may differ from VM because of the divergence of technology.

Recently, many cloud providers such as Amazon EC2 Container Service, Google Container Engine, Docker Datacenter, Rackspace adopt containers in their cloud infrastructure and offer them to clients.

4 Open Issues and Research Challenges

Despite the diverse studies developed about elasticity in cloud computing. There are still many open issues about elasticity in general and research challenges about elasticity in the container emerging technology that the cloud providers and research academy have to deal with.
Open issues about elasticity are:

- **Interoperability**: In order to provide redundancy and ensure reliability, the resources (compute, storage, etc.) should be seamlessly leased from different cloud providers or data centers to the clients. Cloud providers use their own technology and techniques according to their policy, budget, technical skills, etc. Therefore, it is difficult to use multiple clouds to provide resources due to the incompatibilities between them. The combined use of diverse cloud providers remains a challenge because of the lack of standardized APIs, each provider has its own method on how users and applications interact with the cloud infrastructure. It is not only the job of research to solve this challenge, rather the industry needs to agree on standards. Though there are some academic works that allow allocating resources from different providers or data centers, they are limited to certain criteria, for example, [53] allows to allocate resources according to the price offered or spot that matches the user's bid.
- **Granularity**: As seen in Section 2.2.1, IaaS providers offer a fixed set of resources such as Amazon instances, though some users or applications have different needs, as an example, some applications need more CPU than memory. Generally, there must be a coordination in the resource provisioning or de-provisioning. Most of elasticity strategies are based on the horizontal elasticity. Thus, vertical elasticity is very important to provide a related combination of resources according to the demand. There are many academic works [116], [133], [127] which resize CPU, memory or both but there is no coordination between CPU and memory controllers. They resize CPU and memory without regarding the coordination between them. There are just a few works such as [132] which coordinates the provisioning of both resources. In addition to the resource granularity, billing granularity is another issue. Cloud providers charge clients based on the resource consumption per fixed time unit, almost all cloud providers use hour as a minimal billing time unit. For example, using this billing system, VM is billed for an hour even when used for 5 minutes. Few providers CloudSigma [17], VPS.NET [195] allow to use fine-grained billing system where the client will pay approximately its real consumption of resources. The type of the elasticity method has a great impact on the pricing model. For example, implementing vertical elasticity is accompanied

by shifting towards fine-grained pricing policies while using horizontal elasticity leads to extra costs since it uses instances (i.e., VMs) as scaling units (coarse-grained scale) and it also implies running load-balancer (i.e., additional consumption of resources). In addition, container billing is another pricing ambiguity. Since containers are being recently used in production environments, there is no standard pricing model for containers. For example, Amazon charges by VM instance for the Amazon EC2 Container Service. Containers are usually accompanied by orchestrators and cluster of nodes, and the container may settle on VM or on a bare-metal host, therefore, there is still no standard pricing model.

- **Resource availability**: The resource offered by the cloud providers are limited. Therefore, the elasticity of scaling resources is limited by the capacity of the cloud infrastructure. In practice, no cloud provider offers unlimited resources to its clients, but big providers such as Google and Amazon are conceptually unlimited for typical users. However, temporal network bottlenecks, limited geographical locations, higher latency, etc. may hinder the provisioning of resources.
- **Hybrid solutions**: Reactive and proactive approaches have their advantages and drawbacks. Therefore, a sophisticated solution could combine both reactive and proactive approaches and methods such as horizontal and vertical scaling.
- **Start-up time** or spin-up time is defined as the time needed to allocate resources in response to the client demand. Start-up time can reach several minutes but the worse is that the users (clients/customers) are charged directly once they make their requests to scale-up or scale-down resources before acquiring the resource. Provisioning resources may arrive late, and there are chargeable costs, which are different from the real costs that match the provided resources. Start-up time might be fast or slow, it depends on several factors such as cloud layer (IaaS or PaaS), target operating system, number of requested VMs, VM size, resource availability in the region and elasticity mechanism. The lower the start up time is, the better the elastic solution is. Higher start up time affects the efficiency of elasticity system.
- **Thresholds definition**: As we have discussed in Section 2.2.4, threshold-based mechanisms are based on defining thresholds for the measured metrics such as CPU or memory utilization. Choosing suitable thresholds is not an easy task, it is very tricky due to the workload or application behavior changes, that makes the accuracy of the elasticity rules subjective and prone to uncertainty. This can lead to instability of the system. Therefore, it is necessary to have an intelligent self-adaptation system to deal with these uncertainties.
- **Prediction-estimation error**: Proactive techniques anticipate changes in the workload and react in advance to scale-up or scale-down the resources. Herein the start-up time issue is handled using these approaches, however, they could yield errors or what is called prediction-estimation error. Estimation error can lead to resources over-provisioning or under-provisioning. Proactive approaches are characterized as complicated and sophisticated solutions, however, they are not accurate in some cases, and this also depends on the application behavior, unexpected workload changes such as sudden burst or decrease. Some applications are hard to predict, in consequence, predictive techniques can deviate from the intended objectives. Having efficient prediction error handling mechanisms to meet application SLOs with minimum resource cost is worth considering.
- **Optimal trade-off between the user's requirements and provider's interests**: There is a contradiction between provider's profit and user's QoE [61]. Users' QoE is defined as the user satisfaction towards a service. The users search to increase their QoE with the best price and to avoid inadequate provision of resources. While the cloud providers search to increase their profit with providing good QoS services, which means elasticity must ensure better use of computing resources and more energy savings and allows multiple users to be served simultaneously. In addition, due to the market concurrence, cloud providers have to offer cost-effective and QoS-aware services. Therefore, finding an optimal trade-off between user-centric (response time, budget spent, etc.) and provider-centric (reliability, availability, profit) requirements is a big challenge. Offering good QoS will increase customers' satisfaction, this will reflect a good reputation for the provider, and the number of consumers will increase. Hence, the better QoE, the better profits can come from the satisfied customers. Generally, integrating QoE and QoS in the Cloud ecosystem is a promising research domain that is still in its early stages.
- **Unified platforms for elastic applications**: Before discussing elasticity and scalability, the application itself should be elastic. Much of the elasticity solutions implemented by the cloud providers are appropriate for certain types of applications such as server-based applications that depend on the replication of virtual instances and load balancers to distribute the workload among the instances. For that reason, what needed is the development of unified platforms, tools, languages, patterns, abstractions, architectures, etc. to support building and execution of elastic applications. These tools must take into consideration the many application characteristics such as parallelism in order to use elasticity in clouds. Developing such tools, architectures, etc. is a big challenge and worth research, particularly as there is a huge movement towards elasticity and distributed architecture in the computational clouds.
- **Evaluation methodology**: There is no common approach for evaluating elasticity solutions. It is extremely difficult to compare and evaluate different elastic approaches using a formal evaluation technique and a unified testing platform due to the heterogeneity of elastic systems, in addition to the nature of different workload behaviors. In [196], A Performance Evaluation Framework for Auto-Scaling Strategies in Cloud Applications (PEAS) is proposed, however, the framework cannot be generalized on all elastic solutions and evaluation scenarios.

Research challenges about elasticity of containers are:

- **Monitoring containers**: In order to provide data to be analyzed and to make elasticity decisions or actions, monitoring is an essential part in elasticity solutions. However,

it is not an easy task especially with containers. Container holds applications and all of their dependencies and in general many containers may be hosted on the same machine, therefore having stable systems that accurately and rapidly monitor multiple containers is worth searching. In fact, the monitoring challenge is not fully addressed in container technologies.

- **Container-based elasticity**: There are many sophisticated elasticity solutions for the traditional hypervisor-based virtualization. Using these solutions with containers is still an open challenge and research perspective. New container technologies such as Docker use cgroups to limit the resources consumed by a container, such as memory, disk space and I/O, and also offer metrics about these resources. A container can have static resource limits such as 1 CPU and 2G of RAM or can relativity share resources with other containers on the hosting machine. Using the latter technique, the container will get its resources in function of resource usage for the neighboring containers or applications. For some reasons such as cost and priority, static limits are set on containers. The questions which arise are: i) Can we apply the elasticity solutions used in VM on the containers? ii) How to use proactive approaches to anticipate container resource usage and react in advance to scale up/down resources? In addition, many orchestration tools such as Kubernetes, Rancher, etc. are used to manage and orchestrate clusters of containers, but integrating autonomic vertical and horizontal elasticity in these platforms is important.
- **Combined elasticity between VMs and containers**: Nowadays, cloud providers use containers on the top of virtual machines (see Fig. 4). This allows to have many instances arranged across levels of hierarchy. Adjusting container resources such as CPU, RAM, etc. to the demand or workload at runtime will lead to efficient resource utilization, and avoid SLA violations. The problem here is that resizing container resources is limited by the resources of the virtual machine in which it is placed. After certain limits, the container cannot gain more resources, fortunately the VM could be resized by its hypervisor, which by its turn will allow to further resize the container. The challenge to coordinate elasticity between the virtual machine and its placed containers remains unaddressed. Achieving elasticity control for VM and containers will allow a great flexibility and would be an efficient elasticity solution.

5 Related Work

In this section, some of the related works that are relevant to our work are presented. Being the key property behind cloud computing, several works on elasticity are carried out involving various elasticity approaches that depend on the infrastructure, application or workload behavior. [4] is an old survey, it proposes a basic classification for elasticity solutions based on only four characteristics: scope, policy, purpose and method. In addition, the discussion about these characteristics is limited. New characteristics and even new subcategories have appeared in more recent elastic solutions such as the different techniques in workload anticipation in proactive mode. [13] proposes a classification of the techniques for managing elasticity based on strategy and action. The concentration in this paper is on the elasticity strategy. The strategy in this context studies elasticity management solutions based on the quality goal. The quality goal can be the Quality of Business or the Quality of Service from the Cloud Provider (CP) and Application Service Provider (ASP) perspectives. Quality of Business refers to the service provider's revenue/profit, satisfaction. Three solutions are evaluated based on this proposition depending on the strategy adopted and whether reactive or proactive action is followed to achieve elasticity. [152] concentrates mainly on the auto-scaling reactive and proactive approaches and elasticity tools. This work is limited to auto-scaling techniques and experimentation tools. [3] addresses the elasticity definition, metrics and tools. It brought many elasticity definitions, in addition to statistical information about elasticity, such as the number of papers published per year, per country. [14] is another work on cloud elasticity. It is a complementary to our work, but we present elasticity strategies and research challenges in more broader fashion. For example, the mechanisms that can be reactive or proactive, we clearly identified solutions that use these mechanisms in each subcategory. [197] provides a survey of auto-scaling techniques for web applications. According to this work, the actions of auto-scaling systems are based on performance indicators that can be high or low level metrics. Low level metrics such as CPU utilization are performance indicators observed at the server layer while high level metrics such as response time are performance indicators observed at the application layer. This survey is limited to one category of applications, i.e., web applications. Our work differs from the above works in the following aspects: firstly, a complete overview of the mechanisms implemented in the elasticity solutions is provided, an extended classification is proposed including the embedded elasticity. We have described elasticity based on seven characteristics: configuration, scope, purpose, mode, method, provider and architecture. We have further classified each approach into sub mechanisms. For example, time series analysis is a proactive approach that anticipates workloads. It uses many mechanisms: moving average, auto regression, ARMA, holt winter and machine learning; we have provided examples for each case. Secondly, contrary to all previous works, this article is the first that presents works related to container elasticity. Finally, challenges and research perspectives for both VMs and containers are handled in a broader context according to our point of view.

6 Conclusion

Cloud computing is becoming increasingly popular; it is being used extensively by many enterprises with a rapid growing. The key feature that makes cloud platforms attractive is elasticity. Elasticity allows providing elastic resources according to the needs in an optimal way. In this article, a comprehensive study about elasticity is provided. It started by talking about the elasticity definitions, and its related terms scalability and efficiency. We have suggested an extended classification for the elasticity strategies based on the existing academic and commercial solutions. The proposed

classification or taxonomy covers many features and aspects of the cloud elasticity based on the analysis of diverse proposals. Each aspect is then discussed in details providing examples from the proposed proposals that handle cloud elasticity. We have talked about the containerization and the orchestration tools where elasticity will be popular in this new technology. Many works on the container elasticity are presented. Finally, challenges and new research perspectives are presented.

ACKNOWLEDGMENT

This work is supported by the OCCIware research and development project (www.occiware.org) funded by French Programme d'Investissements d'Avenir. Likewise, this work is also funded by Scalair company (www.scalair.fr).

REFERENCES

[1] L. Badger, T. Grance, R. Patt-Corner, and J. Voas, "Draft cloud computing synopsis and recommendations," *NIST special publication*, vol. 800, p. 146, 2011.
[2] C. Pahl, "Containerization and the Paas Cloud," *IEEE Cloud Computing*, vol. 2, no. 3, pp. 24–31, 2015.
[3] E. F. Coutinho, F. R. de Carvalho Sousa, P. A. L. Rego, D. G. Gomes, and J. N. de Souza, "Elasticity in Cloud Computing: a Survey," *Annals of Telecommunications*, pp. 1–21, 2015.
[4] G. Galante and L. C. E. d. Bona, "A Survey on Cloud Computing Elasticity," in *Proceedings of the 2012 IEEE/ACM Fifth International Conference on Utility and Cloud Computing, UCC '12*. Washington, DC, USA: IEEE Computer Society, 2012, pp. 263–270.
[5] N. R. Herbst, S. Kounev, and R. Reussner, "Elasticity in Cloud Computing: What It Is, and What It Is Not," in *Proceedings of the 10th International Conference on Autonomic Computing (ICAC 13)*. San Jose, CA: USENIX, 2013, pp. 23–27.
[6] S. Lehrig, H. Eikerling, and S. Becker, "Scalability, Elasticity, and Efficiency in Cloud Computing: A Systematic Literature Review of Definitions and Metrics," in *Proceedings of the 11th International ACM SIGSOFT Conference on Quality of Software Architectures, QoSA '15*. New York, NY, USA: ACM, 2015, pp. 83–92.
[7] J. Chen, C. Wang, B. B. Zhou, L. Sun, Y. C. Lee, and A. Y. Zomaya, "Tradeoffs Between Profit and Customer Satisfaction for Service Provisioning in the Cloud," in *Proceedings of the 20th International Symposium on High Performance Distributed, HPDC'11*. New York, NY, USA: ACM, 2011, pp. 229–238.
[8] S. Genaud and J. Gossa, "Cost-wait Trade-offs in Client-side Resource Provisioning with Elastic Clouds," in *4th IEEE International Conference on Cloud Computing (CLOUD 2011)*, Washington, United States, Jul. 2011, pp. 1–8.
[9] S. Islam, K. Lee, A. Fekete, and A. Liu, "How a Consumer Can Measure Elasticity for Cloud Platforms," in *Proceedings of the 3rd ACM/SPEC International Conference on Performance Engineering, ICPE '12*. New York, NY, USA: ACM, 2012, pp. 85–96.
[10] M. Kuperberg, N. R. Herbst, J. G. von Kistowski, and R. Reussner, "Defining and Quantifying Elasticity of Resources in Cloud Computing and Scalable Platforms," Karlsruhe Institute of Technology (KIT), Am Fasanengarten 5, 76131 Karlsruhe, Germany, Tech. Rep., 2011.
[11] W. Ai, K. Li, S. Lan, F. Zhang, J. Mei, K. Li, and R. Buyya, "On Elasticity Measurement in Cloud Computing," *Scientific Programming*, vol. 2016, pp. 8–, 2016.
[12] G. Galante and L. C. E. D. Bona, "A programming-level approach for elasticizing parallel scientific applications," *Journal of Systems and Software*, vol. 110, pp. 239 – 252, 2015.
[13] A. Najjar, X. Serpaggi, C. Gravier, and O. Boissier, "Survey of Elasticity Management Solutions in Cloud Computing," in *Continued Rise of the Cloud*. Springer, 2014, pp. 235–263.
[14] A. Naskos, A. Gounaris, and S. Sioutas, "Cloud Elasticity: A Survey," in *Algorithmic Aspects of Cloud Computing*. Springer, 2016, pp. 151–167.
[15] H. Ghanbari, B. Simmons, M. Litoiu, C. Barna, and G. Iszlai, "Optimal Autoscaling in a IaaS Cloud," in *Proceedings of the 9th International Conference on Autonomic Computing, ICAC '12*. New York, NY, USA: ACM, 2012, pp. 173–178.
[16] Profitbricks, Website https://www.profitbricks.com/why-profitbricks#section=cloud-provider-overview.
[17] CloudSigma, Website https://www.cloudsigma.com/features/.
[18] S. C. Nayak and C. Tripathy, "Deadline sensitive lease scheduling in cloud computing environment using AHP," *Journal of King Saud University-Computer and Information Sciences*, 2016.
[19] H. Wang, H. Tianfield, and Q. Mair, "Auction Based Resource Allocation in Cloud Computing," *Multiagent Grid Syst.*, vol. 10, no. 1, pp. 51–66, Jan. 2014.
[20] Google App Engine, Website https://cloud.google.com/appengine/.
[21] Microsoft, Website https://azure.microsoft.com/en-us/.
[22] A. Ashraf, B. Byholm, and I. Porres, "CRAMP: Cost-efficient Resource Allocation for Multiple web applications with Proactive scaling," in *2012 IEEE 4th International Conference on Cloud Computing Technology and Science (CloudCom)*, Dec 2012, pp. 581–586.
[23] H. Fernandez, G. Pierre, and T. Kielmann, "Autoscaling Web Applications in Heterogeneous Cloud Infrastructures," in *2014 IEEE International Conference on Cloud Engineering (IC2E)*, March 2014, pp. 195–204.
[24] S. M.-K. Gueye, N. D. Palma, E. Rutten, A. Tchana, and N. Berthier, "Coordinating self-sizing and self-repair managers for multi-tier systems," *Future Generation Computer Systems*, vol. 35, pp. 14 – 26, 2014.
[25] W. Iqbal, M. N. Dailey, D. Carrera, and P. Janecek, "Adaptive resource provisioning for read intensive multi-tier applications in the cloud," *Future Generation Computer Systems*, vol. 27, no. 6, pp. 871 – 879, 2011.
[26] E. Kalyvianaki, T. Charalambous, and S. Hand, "Self-adaptive and Self-configured CPU Resource Provisioning for Virtualized Servers Using Kalman Filters," in *Proceedings of the 6th International Conference on Autonomic Computing, ICAC '09*. New York, NY, USA: ACM, 2009, pp. 117–126.
[27] P. D. Kaur and I. Chana, "A resource elasticity framework for qos-aware execution of cloud applications," *Future Generation Computer Systems*, vol. 37, pp. 14 – 25, 2014.
[28] F. Paraiso, P. Merle, and L. Seinturier, "soCloud: A Service-Oriented Component-Based PaaS for Managing Portability, Provisioning, Elasticity, and High Availability across Multiple Clouds," *Computing*, pp. 1–27, 2014.
[29] N. Vasić, D. Novaković, S. Miučin, D. Kostić, and R. Bianchini, "DejaVu: Accelerating Resource Allocation in Virtualized Environments," in *Proceedings of the Seventeenth International Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS XVII*. New York, NY, USA: ACM, 2012, pp. 423–436.
[30] Amazon, Website http://aws.amazon.com.
[31] W. Dawoud, I. Takouna, and C. Meinel, "Elastic VM for cloud resources provisioning optimization," in *Advances in Computing and Communications*. Springer, 2011, pp. 431–445.
[32] P. Marshall, K. Keahey, and T. Freeman, "Elastic Site: Using Clouds to Elastically Extend Site Resources," in *2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing (CCGrid)*, May 2010, pp. 43–52.
[33] Rackpace, Website http://www.rackspace.com.
[34] Rightscale, Website http://www.rightscale.com.
[35] N. Roy, A. Dubey, and A. Gokhale, "Efficient autoscaling in the cloud using predictive models for workload forecasting," in *2011 IEEE International Conference on Cloud Computing (CLOUD)*. IEEE, 2011, pp. 500–507.
[36] Scalr, Website http://www.scalr.com.
[37] U. Sharma, P. Shenoy, S. Sahu, and A. Shaikh, "A cost-aware elasticity provisioning system for the cloud," in *2011 31st International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2011, pp. 559–570.
[38] Z. Shen, S. Subbiah, X. Gu, and J. Wilkes, "CloudScale: Elastic Resource Scaling for Multi-tenant Cloud Systems," in *Proceedings of the 2nd ACM Symposium on Cloud Computing, SOCC '11*. New York, NY, USA: ACM, 2011, pp. 5:1–5:14.
[39] A. Iordache, C. Morin, N. Parlavantzas, E. Feller, and P. Riteau, "Resilin: Elastic MapReduce over Multiple Clouds," in *13th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*. Delft, Netherlands: ACM, May 2013, pp. 261–268.
[40] S. Vijayakumar, Q. Zhu, and G. Agrawal, "Dynamic Resource Provisioning for Data Streaming Applications in a Cloud Environment," in *2010 IEEE Second International Conference on Cloud*

Computing Technology and Science (CloudCom), Nov 2010, pp. 441–448.

[41] T. C. Chieu and A. Mohindra and A. A. Karve, "Scalability and Performance of Web Applications in a Compute Cloud," in *2011 IEEE 8th International Conference on e-Business Engineering (ICEBE)*, Oct 2011, pp. 317–323.

[42] N. Grozev and R. Buyya, "Multi-cloud provisioning and load distribution for three-tier applications," *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, vol. 9, no. 3, p. 13, 2014.

[43] P. Sobeslavsky, "Elasticity in Cloud Computing," Ph.D. dissertation, Joseph Fourier University, ENSIMAG, 2011.

[44] X. Zhang, A. Kunjithapatham, S. Jeong, and S. Gibbs, "Towards an Elastic Application Model for Augmenting the Computing Capabilities of Mobile Devices with Cloud Computing," *Mob. Netw. Appl.*, vol. 16, no. 3, pp. 270–284, Jun. 2011.

[45] I. Neamtiu, "Elastic executions from inelastic programs," in *Proceedings of the 6th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*. ACM, 2011, pp. 178–183.

[46] D. Rajan, A. Canino, J. A. Izaguirre, and D. Thain, "Converting a High Performance Application to an Elastic Cloud Application," in *2011 IEEE Third International Conference on Cloud Computing Technology and Science (CloudCom)*, Nov 2011, pp. 383–390.

[47] S. Moller and A. Raake, "Quality of experience," *New York, US: Springer*, 2014.

[48] T. Hobfeld and R. Schatz and M. Varela and C. Timmerer, "Challenges of QoE Management for Cloud Applications," *IEEE Communications Magazine*, vol. 50, no. 4, pp. 28–36, April 2012.

[49] F. D. Munoz-Escoı and J. M. Bernabeu-Aubán, "A Survey on Elasticity Management in the PaaS Service Model," *Technical Report ITI-SIDI-2015/002, Universitat Politecnica de Valencia, Spain*.

[50] R. N. Calheiros, C. Vecchiola, D. Karunamoorthy, and R. Buyya, "The Aneka platform and QoS-driven resource provisioning for elastic applications on hybrid Clouds ," *Future Generation Computer Systems*, vol. 28, no. 6, pp. 861 – 870, 2012.

[51] S. Dutta, S. Gera, A. Verma, and B. Viswanathan, "SmartScale: Automatic Application Scaling in Enterprise Clouds," in *2012 IEEE 5th International Conference on Cloud Computing (CLOUD)*, June 2012, pp. 221–228.

[52] AWS Spot Instances, Website https://aws.amazon.com/ec2/spot/.

[53] R. Buyya, R. Ranjan, and R. N. Calheiros, "InterCloud: Utility-oriented federation of cloud computing environments for scaling of application services," in *Algorithms and architectures for parallel processing*. Springer, 2010, pp. 13–31.

[54] A. da Silva Dias, L. H. V. Nakamura, J. C. Estrella, R. H. C. Santana, and M. J. Santana, "Providing IaaS resources automatically through prediction and monitoring approaches," in *2014 IEEE Symposium on Computers and Communication (ISCC)*, June 2014, pp. 1–7.

[55] R. Han, M. M. Ghanem, L. Guo, Y. Guo, and M. Osmond, "Enabling cost-aware and adaptive elasticity of multi-tier cloud applications," *Future Generation Computer Systems*, vol. 32, pp. 82–98, 2014.

[56] P. Hoenisch, I. Weber, S. Schulte, L. Zhu, and A. Fekete, "Four-Fold Auto-Scaling on a Contemporary Deployment Platform Using Docker Containers," in *Service-Oriented Computing*. Springer, 2015, pp. 316–323.

[57] D. Perez-Palacin, R. Mirandola, and R. Calinescu, "Synthesis of Adaptation Plans for Cloud Infrastructure with Hybrid Cost Models," in *Proceedings of the 2014 40th EUROMICRO Conference on Software Engineering and Advanced Applications, SEAA '14*. Washington, DC, USA: IEEE Computer Society, 2014, pp. 443–450.

[58] L. M. Vaquero, D. Morán, F. Galán, and J. M. Alcaraz-Calero, "Towards Runtime Reconfiguration of Application Control Policies in the Cloud," *Journal of Network and Systems Management*, vol. 20, no. 4, pp. 489–512, Dec. 2012.

[59] D. Villela, P. Pradhan, and D. Rubenstein, "Provisioning servers in the application tier for e-commerce systems," in *Twelfth IEEE International Workshop on Quality of Service, IWQOS 2004*, June 2004, pp. 57–66.

[60] A. Beloglazov and R. Buyya, "Adaptive Threshold-based Approach for Energy-efficient Consolidation of Virtual Machines in Cloud Data Centers," in *Proceedings of the 8th International Workshop on Middleware for Grids, Clouds and e-Science, MGC '10*. New York, NY, USA: ACM, 2010, pp. 4:1–4:6.

[61] H.-J. Hong, D.-Y. Chen, C.-Y. Huang, K.-T. Chen, and C.-H. Hsu, "Placing virtual machines to optimize cloud gaming experience," *IEEE Transactions on Cloud Computing*, vol. 3, no. 1, pp. 42–53, 2015.

[62] K. Keahey, P. Armstrong, J. Bresnahan, D. LaBissoniere, and P. Riteau, "Infrastructure Outsourcing in Multi-cloud Environment," in *Proceedings of the 2012 Workshop on Cloud Services, Federation, and the 8th Open Cirrus Summit, FederatedClouds '12*. New York, NY, USA: ACM, 2012, pp. 33–38.

[63] F. Paraiso, P. Merle, and L. Seinturier, "Managing Elasticity Across Multiple Cloud Providers," in *Proceedings of the 2013 International Workshop on Multi-cloud Applications and Federated Clouds, MultiCloud '13*. New York, NY, USA: ACM, 2013, pp. 53–60.

[64] E. Kafetzakis and H. Koumaras and M. A. Kourtis and V. Koumaras, "QoE4CLOUD: A QoE-driven multidimensional framework for cloud environments," in *2012 International Conference on Telecommunications and Multimedia (TEMU)*, July 2012, pp. 77–82.

[65] A. Najjar, C. Gravier, X. Serpaggi, and O. Boissier, "Modeling User Expectations Satisfaction for SaaS Applications Using Multi-agent Negotiation," in *2016 IEEE/WIC/ACM International Conference on Web Intelligence (WI)*, Oct 2016, pp. 399–406.

[66] J. Hadley, Y. El Khatib, G. Blair, and U. Roedig, *MultiBox: lightweight containers for vendor-independent multi-cloud deployments*, ser. Communications in Computer and Information Science. Springer Verlag, 11 2015, pp. 79–90.

[67] C. Kan, "DoCloud: An elastic cloud platform for Web applications based on Docker," in *2016 18th International Conference on Advanced Communication Technology (ICACT)*, Jan 2016, pp. 478–483.

[68] P. P. Kukade and G. Kale, "Auto-Scaling of Micro-Services Using Containerization," *International Journal of Science and Research (IJSR)*, pp. 1960–1963, 2013.

[69] F. Paraiso, S. Challita, Y. Al-Dhuraibi, and P. Merle, "Model-Driven Management of Docker Containers," in *9th IEEE International Conference on Cloud Computing (CLOUD)*, San Francisco, United States, Jun. 2016, pp. 718–725.

[70] Proxmox VE, Website https://www.proxmox.com/en/proxmox-ve.

[71] AzureWatch, Website http://www.paraleap.com/azurewatch/.

[72] E. Barrett, E. Howley, and J. Duggan, "Applying reinforcement learning towards automating resource allocation and application scalability in the cloud," *Concurrency and Computation: Practice and Experience*, vol. 25, no. 12, pp. 1656–1674, 2013.

[73] T. Knauth and C. Fetzer, "Scaling Non-elastic Applications Using Virtual Machines," in *2011 IEEE International Conference on Cloud Computing (CLOUD)*, July 2011, pp. 468–475.

[74] G. Moltó, M. Caballer, E. Romero, and C. de Alfonso, "Elastic memory management of virtualized infrastructures for applications with dynamic memory requirements," *Procedia Computer Science*, vol. 18, pp. 159–168, 2013.

[75] G. A. Moreno, J. Cmara, D. Garlan, and B. Schmerl, "Efficient Decision-Making under Uncertainty for Proactive Self-Adaptation," in *2016 IEEE International Conference on Autonomic Computing (ICAC)*, July 2016, pp. 147–156.

[76] S. Barker, Y. Chi, H. Hacigümüs, P. Shenoy, and E. Cecchet, "ShuttleDB: Database-Aware Elasticity in the Cloud," in *11th International Conference on Autonomic Computing (ICAC 14)*. Philadelphia, PA: USENIX Association, Jun. 2014, pp. 33–43.

[77] L. Beernaert, M. Matos, R. Vilaça, and R. Oliveira, "Automatic Elasticity in OpenStack," in *Proceedings of the Workshop on Secure and Dependable Middleware for Cloud Monitoring and Management, SDMCMM '12*. New York, NY, USA: ACM, 2012, pp. 2:1–2:6.

[78] F. Cruz, F. Maia, M. Matos, R. Oliveira, J. a. Paulo, J. Pereira, and R. Vilaça, "MeT: Workload Aware Elasticity for NoSQL," in *Proceedings of the 8th ACM European Conference on Computer Systems, EuroSys '13*. New York, NY, USA: ACM, 2013, pp. 183–196.

[79] E. Kassela, C. Boumpouka, I. Konstantinou, and N. Koziris, "Automated workload-aware elasticity of NoSQL clusters in the cloud," in *2014 IEEE International Conference on Big Data (Big Data)*, Oct 2014, pp. 195–200.

[80] R. Han, L. Guo, M. Ghanem, and Y. Guo, "Lightweight Resource Scaling for Cloud Applications," in *2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, May 2012, pp. 644–651.

[81] P. Jamshidi, A. Ahmad, and C. Pahl, "Autonomic resource provisioning for cloud-based software," in *Proceedings of the 9th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*. ACM, 2014, pp. 95–104.

[82] I. Konstantinou, E. Angelou, D. Tsoumakos, C. Boumpouka, N. Koziris, and S. Sioutas, "TIRAMOLA: Elastic NoSQL Provisioning Through a Cloud Management Platform," in *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data, SIGMOD '12*. New York, NY, USA: ACM, 2012, pp. 725–728.

[83] L. R. Moore, K. Bean, and T. Ellahi, "A coordinated reactive and predictive approach to cloud elasticity," *CLOUD COMPUTING 2013: The Fourth International Conference on Cloud Computing, GRIDs, and Virtualization*, pp. 87–92, 2013.

[84] Moore, Laura R. and Bean, Kathryn and Ellahi, Tariq, "Transforming Reactive Auto-scaling into Proactive Auto-scaling," in *Proceedings of the 3rd International Workshop on Cloud Data and Platforms, CloudDP '13*. New York, NY, USA: ACM, 2013, pp. 7–12.

[85] F. J. A. Morais, F. V. Brasileiro, R. V. Lopes, R. A. Santos, W. Satterfield, and L. Rosa, "Autoflex: Service Agnostic Auto-scaling Framework for IaaS Deployment Models," in *2013 13th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, May 2013, pp. 42–49.

[86] H. Nguyen, Z. Shen, X. Gu, S. Subbiah, and J. Wilkes, "Agile: Elastic distributed resource scaling for infrastructure-as-a-service," in *Proceedings of the USENIX International Conference on Automated Computing (ICAC13). San Jose, CA*, 2013, pp. 69–82.

[87] P. D. Sanzo, D. Rughetti, B. Ciciani, and F. Quaglia, "Auto-tuning of Cloud-Based In-Memory Transactional Data Grids via Machine Learning," in *2012 Second Symposium on Network Cloud Computing and Applications (NCCA)*, Dec 2012, pp. 9–16.

[88] M. Serafini, E. Mansour, A. Aboulnaga, K. Salem, T. Rafiq, and U. F. Minhas, "Accordion: Elastic scalability for database systems supporting distributed transactions," *Proceedings of the VLDB Endowment*, vol. 7, no. 12, pp. 1035–1046, 2014.

[89] D. Serrano, S. Bouchenak, Y. Kouki, T. Ledoux, J. Lejeune, J. Sopena, L. Arantes, and P. Sens, "Towards QoS-Oriented SLA Guarantees for Online Cloud Services," in *2013 13th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, May 2013, pp. 50–57.

[90] Y. Tan, H. Nguyen, Z. Shen, X. Gu, C. Venkatramani, and D. Rajan, "Prepare: Predictive performance anomaly prevention for virtualized cloud systems," in *2012 IEEE 32nd International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2012, pp. 285–294.

[91] L. Baresi, S. Guinea, A. Leva, and G. Quattrocchi, "A Discrete-time Feedback Controller for Containerized Cloud Applications," in *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering, FSE 2016*. New York, NY, USA: ACM, 2016, pp. 217–228.

[92] G. A. Moreno, J. Cámara, D. Garlan, and B. Schmerl, "Proactive Self-adaptation Under Uncertainty: A Probabilistic Model Checking Approach," in *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering, ESEC/FSE 2015*. New York, NY, USA: ACM, 2015, pp. 1–12.

[93] A. Najjar, X. Serpaggi, C. Gravier, and O. Boissier, "Multi-agent Negotiation for User-centric Elasticity Management in the Cloud," in *2013 IEEE/ACM 6th International Conference on Utility and Cloud Computing*, Dec 2013, pp. 357–362.

[94] B. An, V. Lesser, D. Irwin, and M. Zink, "Automated Negotiation with Decommitment for Dynamic Resource Allocation in Cloud Computing," in *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: Volume 1, AAMAS '10*. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2010, pp. 981–988.

[95] T. Chieu, A. Mohindra, A. Karve, and A. Segal, "Dynamic Scaling of Web Applications in a Virtualized Cloud Computing Environment," in *IEEE International Conference on e-Business Engineering ICEBE '09*, Oct 2009, pp. 281–286.

[96] X. Dutreilh, N. Rivierre, A. Moreau, J. Malenfant, and I. Truck, "From Data Center Resource Allocation to Control Theory and Back," in *2010 IEEE 3rd International Conference on Cloud Computing (CLOUD)*, July 2010, pp. 410–417.

[97] M. Z. Hasan, E. Magana, A. Clemm, L. Tucker, and S. L. D. Gudreddi, "Integrated and autonomic cloud resource scaling," in *Network Operations and Management Symposium (NOMS)*. IEEE, 2012, pp. 1327–1334.

[98] M. Maurer, I. Brandic, and R. Sakellariou, "Enacting SLAs in Clouds Using Rules," in *Proceedings of the 17th International Conference on Parallel Processing, Euro-Par'11 - Volume Part I*. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 455–466.

[99] A. Naskos, E. Stachtiari, A. Gounaris, P. Katsaros, D. Tsoumakos, I. Konstantinou, and S. Sioutas, "Dependable Horizontal Scaling Based on Probabilistic Model Checking," in *2015 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, May 2015, pp. 31–40.

[100] H. C. Lim, S. Babu, and J. S. Chase, "Automated Control for Elastic Storage," in *Proceedings of the 7th International Conference on Autonomic Computing, ICAC '10*. New York, NY, USA: ACM, 2010, pp. 1–10.

[101] H. C. Lim, S. Babu, J. S. Chase, and S. S. Parekh, "Automated Control in Cloud Computing: Challenges and Opportunities," in *Proceedings of the 1st Workshop on Automated Control for Datacenters and Clouds, ACDC '09*. New York, NY, USA: ACM, 2009, pp. 13–18.

[102] X. Dutreilh, S. Kirgizov, O. Melekhova, J. Malenfant, N. Rivierre, and I. Truck, "Using reinforcement learning for autonomic resource allocation in clouds: Towards a fully automated workflow," in *Seventh International Conference on Autonomic and Autonomous Systems ICAS*, 2011, pp. 67–74.

[103] B. Urgaonkar, P. Shenoy, A. Chandra, P. Goyal, and T. Wood, "Agile Dynamic Provisioning of Multi-tier Internet Applications," *ACM Transactions on Autonomous and Adaptive Systems*, vol. 3, no. 1, pp. 1:1–1:39, Mar. 2008.

[104] A. Al-Shishtawy and V. Vlassov, "ElastMan: Elasticity Manager for Elastic Key-value Stores in the Cloud," in *Proceedings of the 2013 ACM Cloud and Autonomic Computing Conference, CAC '13*. New York, NY, USA: ACM, 2013, pp. 7:1–7:10.

[105] S.-M. Park and M. Humphrey, "Self-Tuning Virtual Machines for Predictable eScience," in *Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, CCGRID '09*. Washington, DC, USA: IEEE Computer Society, 2009, pp. 356–363.

[106] J. Huang, C. Li, and J. Yu, "Resource prediction based on double exponential smoothing in cloud computing," in *2nd International Conference on Consumer Electronics Communications and Networks (CECNet)*, April 2012, pp. 2056–2060.

[107] H. Mi, H. Wang, G. Yin, Y. Zhou, D. Shi, and L. Yuan, "Online Self-Reconfiguration with Performance Guarantee for Energy-Efficient Large-Scale Cloud Computing Data Centers," in *2010 IEEE International Conference on Services Computing (SCC)*, July 2010, pp. 514–521.

[108] Z. Gong, X. Gu, and J. Wilkes, "Press: Predictive elastic resource scaling for cloud systems," in *2010 International Conference on Network and Service Management (CNSM)*, Oct 2010, pp. 9–16.

[109] S. Khatua, A. Ghosh, and N. Mukherjee, "Optimizing the utilization of virtual resources in Cloud environment," in *2010 IEEE International Conference on Virtual Environments Human-Computer Interfaces and Measurement Systems (VECIMS)*, Sept 2010, pp. 82–87.

[110] S. Islam, J. Keung, K. Lee, and A. Liu, "Empirical Prediction Models for Adaptive Resource Provisioning in the Cloud," *Future Generation Computer Systems*, vol. 28, no. 1, pp. 155–162, Jan. 2012.

[111] J. Kupferman, J. Silverman, P. Jara, and J. Browne, "Scaling into the cloud," *CS270-advanced operating systems*, 2009.

[112] E. Caron, F. Desprez, and A. Muresan, "Forecasting for Cloud computing on-demand resources based on pattern matching," INRIA, Research Report RR-7217, Jul. 2010.

[113] J. Rao, X. Bu, C.-Z. Xu, L. Wang, and G. Yin, "VCONF: A Reinforcement Learning Approach to Virtual Machines Auto-configuration," in *Proceedings of the 6th International Conference on Autonomic Computing, ICAC '09*. New York, NY, USA: ACM, 2009, pp. 137–146.

[114] G. Tesauro, N. Jong, R. Das, and M. Bennani, "A Hybrid Reinforcement Learning Approach to Autonomic Resource Allocation," in *IEEE International Conference on Autonomic Computing ICAC '06*, June 2006, pp. 65–73.

[115] A. Ali-Eldin, J. Tordsson, and E. Elmroth, "An adaptive hybrid elasticity controller for cloud infrastructures," in *2012 IEEE Network Operations and Management Symposium (NOMS)*, April 2012, pp. 204–212.

[116] E. Lakew, C. Klein, F. Hernandez-Rodriguez, and E. Elmroth, "Towards faster response time models for vertical elasticity," in *2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing (UCC)*, Dec 2014, pp. 560–565.

[117] Q. Zhang, L. Cherkasova, and E. Smirni, "A Regression-Based Analytic Model for Dynamic Resource Provisioning of Multi-Tier Applications," in *Fourth International Conference on Autonomic Computing ICAC '07*, June 2007, pp. 27–27.

[118] L. Wang, J. Xu, M. Zhao, Y. Tu, and J. Fortes, "Fuzzy Modeling Based Resource Management for Virtualized Database Systems," in *2011 IEEE 19th International Symposium on Modeling, Analysis Simulation of Computer and Telecommunication Systems (MASCOTS)*, July 2011, pp. 32–42.

[119] R. Calheiros, R. Ranjan, and R. Buyya, "Virtual Machine Provisioning Based on Analytical Performance and QoS in Cloud Computing Environments," in *2011 International Conference on Parallel Processing (ICPP)*, Sept 2011, pp. 295–304.

[120] C.-L. Hung, Y.-C. Hu, and K.-C. Li, "Auto-Scaling Model for Cloud Computing System," *International Journal of Hybrid Information Technology*, vol. 5, no. 2, pp. 181–186, 2012.

[121] P. Leitner, W. Hummer, B. Satzger, C. Inzinger, and S. Dustdar, "Cost-Efficient and Application SLA-Aware Client Side Request Scheduling in an Infrastructure-as-a-Service Cloud," in *2012 IEEE 5th International Conference on Cloud Computing (CLOUD)*, June 2012, pp. 213–220.

[122] Z. Liu, S. Wang, Q. Sun, H. Zou, and F. Yang, "Cost-aware cloud service request scheduling for SaaS providers," *The Computer Journal*, pp. 291–301, 2013.

[123] D. Niu, H. Xu, B. Li, and S. Zhao, "Quality-assured cloud bandwidth auto-scaling for video-on-demand applications," in *2012 Proceedings IEEE INFOCOM*, March 2012, pp. 460–468.

[124] J. Tirado, D. Higuero, F. Isaila, and J. Carretero, "Predictive data grouping and placement for cloud-based elastic server infrastructures," in *2011 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, May 2011, pp. 285–294.

[125] S. Spinner, S. Kounev, X. Zhu, L. Lu, M. Uysal, A. Holler, and R. Griffith, "Runtime Vertical Scaling of Virtualized Applications via Online Model Estimation," in *2014 IEEE Eighth International Conference on Self-Adaptive and Self-Organizing Systems (SASO)*, Sept 2014, pp. 157–166.

[126] L. Yazdanov and C. Fetzer, "Vertical Scaling for Prioritized VMs Provisioning," in *2012 Second International Conference on Cloud and Green Computing (CGC)*, Nov 2012, pp. 118–125.

[127] Y. Wang, C. C. Tan, and N. Mi, "Using Elasticity to Improve Inline Data Deduplication Storage Systems," in *Proceedings of the 2014 IEEE International Conference on Cloud Computing, CLOUD '14*. Washington, DC, USA: IEEE Computer Society, 2014, pp. 785–792.

[128] W. Zhao and Z. Wang, "Dynamic Memory Balancing for Virtual Machines," in *Proceedings of the 2009 ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments, VEE '09*. New York, NY, USA: ACM, 2009, pp. 21–30.

[129] W. Dawoud, I. Takouna, and C. Meinel, "Elastic virtual machine for fine-grained cloud resource provisioning," in *Global Trends in Computing and Communication Systems*. Springer, 2012, pp. 11–25.

[130] Y. Diao, N. Gandhi, J. Hellerstein, S. Parekh, and D. Tilbury, "Using mimo feedback control to enforce policies for interrelated metrics with application to the apache web server," in *Network Operations and Management Symposium, 2002. NOMS 2002. 2002 IEEE/IFIP*, 2002, pp. 219–234.

[131] Dynamic scaling of CPU and RAM, Website https://cwiki.apache.org/confluence/display/CLOUDSTACK/Dynamic+scaling+of+CPU+and+RAM, Visited 2016/02.

[132] S. Farokhi, E. Lakew, C. Klein, I. Brandic, and E. Elmroth, "Coordinating CPU and Memory Elasticity Controllers to Meet Service Response Time Constraints," in *2015 International Conference on Cloud and Autonomic Computing (ICCAC)*, Sept 2015, pp. 69–80.

[133] L. Lu, X. Zhu, R. Griffith, P. Padala, A. Parikh, P. Shah, and E. Smirni, "Application-driven dynamic vertical scaling of virtual machines in resource pools," in *Network Operations and Management Symposium (NOMS), 2014 IEEE*, May 2014, pp. 1–9.

[134] H. Jin, L. Deng, S. Wu, X. Shi, and X. Pan, "Live virtual machine migration with adaptive, memory compression," in *2009 IEEE International Conference on Cluster Computing and Workshops, CLUSTER '09*, Aug 2009, pp. 1–10.

[135] X. Qin, W. Wang, W. Zhang, J. Wei, X. Zhao, and T. Huang, "Elasticat: A load rebalancing framework for cloud-based key-value stores," in *2012 19th International Conference on High Performance Computing (HiPC)*, Dec 2012, pp. 1–10.

[136] R. S. S. Kirthica1, "Provisioning rapid elasticity by light-weight live resource migration," *International Journal of Modern Trends in Engineering and Research*, pp. 99–106, July 2015.

[137] Y. Zhao and W. Huang, "Adaptive Distributed Load Balancing Algorithm Based on Live Migration of Virtual Machines in Cloud," in *Proceedings of the 2009 Fifth International Joint Conference on INC, IMS and IDC, NCM '09*. Washington, DC, USA: IEEE Computer Society, 2009, pp. 170–175.

[138] D. Bruneo, "A Stochastic Model to Investigate Data Center Performance and QoS in IaaS Cloud Computing Systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 3, pp. 560–569, March 2014.

[139] S. Das, S. Nishimura, D. Agrawal, and A. El Abbadi, "Albatross: Lightweight Elasticity in Shared Storage Databases for the Cloud Using Live Data Migration," *Proc. VLDB Endowment*, vol. 4, no. 8, pp. 494–505, May 2011.

[140] S. He, L. Guo, and Y. Guo, "Real Time Elastic Cloud Management for Limited Resources," in *2011 IEEE International Conference on Cloud Computing (CLOUD)*, July 2011, pp. 622–629.

[141] M. R. Hines and K. Gopalan, "Post-copy based live virtual machine migration using adaptive pre-paging and dynamic self-ballooning," in *Proceedings of the 2009 ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments*. ACM, 2009, pp. 51–60.

[142] S. Sotiriadis, N. Bessis, C. Amza, and R. Buyya, "Elastic load balancing for dynamic virtual machine reconfiguration based on vertical and horizontal scaling," *IEEE Transactions on Services Computing*, 2016.

[143] N. M. Calcavecchia, B. A. Caprarescu, E. Di Nitto, D. J. Dubois, and D. Petcu, "DEPAS: a decentralized probabilistic algorithm for auto-scaling," *Computing*, vol. 94, no. 8-10, pp. 701–730, 2012.

[144] T. Chieu and H. Chan, "Dynamic Resource Allocation via Distributed Decisions in Cloud Environment," in *8th IEEE International Conference on e-Business Engineering (ICEBE)*, Oct 2011, pp. 125–130.

[145] M. Siebenhaar, T. A. B. Nguyen, U. Lampe, D. Schuller, and R. Steinmetz, "Concurrent Negotiations in Cloud-based Systems," in *Proceedings of the 8th International Conference on Economics of Grids, Clouds, Systems, and Services, GECON'11*. Berlin, Heidelberg: Springer-Verlag, 2012, pp. 17–31.

[146] S. Son and K. M. Sim, "A price-and-time-slot-negotiation mechanism for cloud service reservations," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 42, no. 3, pp. 713–728, 2012.

[147] G. Copil, D. Moldovan, H. L. Truong, and S. Dustdar, "On Controlling Cloud Services Elasticity in Heterogeneous Clouds," in *2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing (UCC)*, Dec 2014, pp. 573–578.

[148] Datapipe, Website https://www.datapipe.com/gogrid/.

[149] P. Marshall, H. Tufo, and K. Keahey, "Provisioning policies for elastic computing environments," in *2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW)*. IEEE, 2012, pp. 1085–1094.

[150] L. Wu, S. K. Garg, and R. Buyya, "SLA-based admission control for a Software-as-a-Service provider in Cloud computing environments," *Journal of Computer and System Sciences*, vol. 78, no. 5, pp. 1280–1299, 2012.

[151] T. L.-B. J. Miguel-Alonso and J. A. Lozano, "A Review of Auto-scaling Techniques for Elastic Applications in Cloud Environments," *Journal Grid Computing*, vol. 12, no. 4, pp. 559–592, Dec. 2014.

[152] T. Lorido-Botrán, J. Miguel-Alonso, and J. A. Lozano, "Auto-scaling techniques for elastic applications in cloud environments," *Department of Computer Architecture and Technology, University of Basque Country, Tech. Rep. EHU-KAT-IK-09*, vol. 12, p. 2012, 2012.

[153] Y. Kuno, K. Nii, and S. Yamaguchi, "A Study on Performance of Processes in Migrating Virtual Machines," in *2011 Tenth International Symposium on Autonomous Decentralized Systems*, March 2011, pp. 567–572.

[154] D. Kapil, E. S. Pilli, and R. C. Joshi, "Live virtual machine migration techniques: Survey and research challenges," in *2013 IEEE 3rd International Advance Computing Conference (IACC)*, Feb 2013, pp. 963–969.

[155] D. G. V V.Vinothina Dr.R.Sridaran, "A Survey on Resource Allocation Strategies in Cloud Computing," *International Journal of Advanced Computer Science and Applications (IJACSA)*, vol. 3, no. 6, pp. 97–104, 2012.
[156] M. G. Hafez and M. S. Elgamel, "Agent-Based Cloud Computing: A Survey," in *2016 IEEE 4th International Conference on Future Internet of Things and Cloud (FiCloud)*, Aug 2016, pp. 285–292.
[157] D. Talia, "Clouds Meet Agents: Toward Intelligent Cloud Services," *IEEE Internet Computing*, vol. 16, no. 2, pp. 78–81, March 2012.
[158] CloudSim, Website http://www.cloudbus.org/cloudsim/.
[159] S. F. Piraghaj, A. V. Dastjerdi, R. N. Calheiros, and R. Buyya, "ContainerCloudSim: An Environment for Modeling and Simulation of Containers in Cloud Data Centers," *Software: Practice and Experience*, 2016.
[160] GreenCloud, Website https://greencloud.gforge.uni.lu.
[161] OMNeT++, Website https://omnetpp.org/.
[162] iCanCloud, Website http://www.arcos.inf.uc3m.es/~icancloud/Home.html.
[163] SimGrid, Website http://simgrid.gforge.inria.fr.
[164] EMUSIM, Website http://www.cloudbus.org/cloudsim/emusim/.
[165] World Cup 98 Trace, Website http://ita.ee.lbl.gov/html/contrib/WorldCup.html.
[166] ClarkNet, Website http://ita.ee.lbl.gov/html/contrib/ClarkNet-HTTP.html.
[167] K. Bangari and C. C. Rao, "Real Workload Characterization and Synthetic Workload Generation."
[168] A. Ali-Eldin, J. Tordsson, E. Elmroth, and M. Kihl, "Workload classification for efficient auto-scaling of cloud resources," Technical Report, 2005.[Online]. Available: www8.cs.umu.se/research/uminf/reports/2013/013/part1.pdf, Tech. Rep., 2005.
[169] Rubbos, Website http://jmob.ow2.org/rubbos.html.
[170] Rubis, Website http://rubis.ow2.org.
[171] TCP-W, Website http://www.tpc.org/tpcw/.
[172] W. Sobel, S. Subramanyam, A. Sucharitakul, J. Nguyen, H. Wong, A. Klepchukov, S. Patil, A. Fox, and D. Patterson, "CloudStone: Multi-platform, multi-language benchmark and measurement tools for Web 2.0," in *Proc. of CCA*, vol. 8, 2008.
[173] YCSB, Website https://en.wikipedia.org/wiki/YCSB.
[174] A. Sangroya, D. Serrano, and S. Bouchenak, "Benchmarking Dependability of MapReduce Systems," in *2012 IEEE 31st Symposium on Reliable Distributed Systems (SRDS)*, Oct 2012, pp. 21–30.
[175] fio, Website http://freecode.com/projects/fio.
[176] CloudSuite, Website http://cloudsuite.ch.
[177] S. Huang, J. Huang, J. Dai, T. Xie, and B. Huang, *The HiBench Benchmark Suite: Characterization of the MapReduce-Based Data Analysis*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 209–228.
[178] Containers: The Next Big Thing in Cloud?, Website http://insights.wired.com/profiles/blogs/what-s-the-next-big-thing-in-cloud#axzz3sJ2gJP26.
[179] W. Felter, A. Ferreira, R. Rajamony, and J. Rubio, "An updated performance comparison of virtual machines and Linux containers," in *International Symposium on IEEE Performance Analysis of Systems and Software (ISPASS)*, March 2015, pp. 171–172.
[180] A. Tosatto, P. Ruiu, and A. Attanasio, "Container-based orchestration in cloud: State of the art and challenges," in *2015 Ninth International Conference on Complex, Intelligent, and Software Intensive Systems (CISIS)*. IEEE, 2015, pp. 70–75.
[181] D. Merkel, "Docker: Lightweight Linux Containers for Consistent Development and Deployment," Houston, TX, Mar. 2014.
[182] An Introduction to CoreOS System Components, Website https://www.digitalocean.com/community/tutorials/an-introduction-to-coreos-system-components.
[183] LXC: What Is LXC, Website https://linuxcontainers.org/lxc/introduction/.
[184] LXD, Website https://linuxcontainers.org/lxd/introduction/.
[185] BSDJails, Website https://www.freebsd.org/cgi/man.cgi?query=jail&format=html.
[186] OpenVZ, Website https://openvz.org/Main_Page.
[187] The Container Ecosystem Project, Website https://sysdig.com/the-container-ecosystem-project/.
[188] Kubernetes, Website http://kubernetes.io/.
[189] Swarm v. Fleet v. Kubernetes v. Mesos, Website http://radar.oreilly.com/2015/10/swarm-v-fleet-v-kubernetes-v-mesos.html.
[190] B. Burns, B. Grant, D. Oppenheimer, E. Brewer, and J. Wilkes, "Borg, Omega, and Kubernetes," *Commun. ACM*, vol. 59, no. 5, pp. 50–57, Apr. 2016.
[191] Containers: Package, ship and run any software as a self-sufficient unit, Website https://coreos.com/using-coreos/containers/.
[192] Apache Mesos, Website http://mesos.apache.org.
[193] Polo Sony, Ivin, "Inter-Cloud application migration and portability using Linux containers for better resource provisioning and interoperability," Ph.D. dissertation, Dublin, National College of Ireland, 2015.
[194] C. Yang, "Checkpoint and Restoration of Micro-service in Docker Containers," *icmii-15*, 2015.
[195] VPS.NET, Website https://www.vps.net.
[196] A. V. Papadopoulos, A. Ali-Eldin, K.-E. Arzen, J. Tordsson, and E. Elmroth, "PEAS: A Performance Evaluation Framework for Auto-Scaling Strategies in Cloud Applications," *ACM Trans. Model. Perform. Eval. Comput. Syst.*, vol. 1, no. 4, pp. 15:1–15:31, Aug. 2016.
[197] C. Qu, R. N. Calheiros, and R. Buyya, "Auto-scaling Web Applications in Clouds: A Taxonomy and Survey," *arXiv preprint arXiv:1609.09224*, 2016.

Yahya Al-dhuraibi received his bachelor of science degree in computer science and engineering in 2010 from Aden University, Yemen. He also received his Master degree in network system architecture from University of Evry, France in 2014. His current research focuses on cloud computing elasticity in Scalair company and Inria labs - University of Lille, France in context of PhD.

Fawaz Paraiso is postdoctoral researcher in Computer Science Department at Inria since 2015. He received his PhD degree in computer science in 2014 from University of Lille 1. Before he joined Inria in 2011, he worked in software and telecommunication companies as software engineer and then lead software engineer. His current research interests include cloud computing, parallel and distributed systems, adaptive systems, middleware, and natural language processing.

Nabil Bashir received his Ph.D degree in computer science from the University of Rennes, France. He also received a diploma of computer science engineering from Batna University, Algeria and M.S. degree in computer science from the University of Rennes, France. His research interests include optimization, algorithms for constrained path computation, architectures and inter-carrier service delivery with assured QoS, complex event processing within Alcatel-Lucent Bell Labs and Inria. Nowadays, he is head of R&D and new technologies department at Scalair cloud provider. Current research is around cloud elasticity and resources optimization.

Philippe Merle is senior researcher at Inria since 2002 and is member of the Spirals research team. He was associate professor at University of Lille 1, France. He obtained a PhD degree in computer science from University of Lille 1. His research is about software engineering for distributed systems, especially cloud computing, service oriented computing, middleware, model driven engineering, and component-based software engineering. He has co-authored two patents, two OMG specifications, one book, 15 journal papers, and more than 70 international conference papers.